Vol. 27 No. 6 情報処理学会論文誌 June 1986

プログラム設計用言語 SL の開発と評価†

稲田 満†† 岡本 務††
渡辺 敏†† 中村雄三††

プログラムの設計仕様を入力し COBOL プログラムを自動生成するプログラム設計用言語 SL (Superb Data Oriented Language) について報告する．SL の設計目的は大規模プログラムへの適用も可能な適用性の広いプログラムジェネレータの実現であり，その特徴は以下のとおりである．①実務的で均質な設計結果の得られることが特色のデータ構造設計法をベースにその生産物である設計仕様書の記述法をデータベースも扱えるように発展させて言語仕様化し，その言語で設計した仕様書からプログラムを自動生成する，②プログラムの入出力処理，業務用関数等の特殊処理に対応する目的プログラムのコードは利用システムごとに微妙な差異があるため固定化せず利用者が定義できるようにしており，利用システム固有の環境に対する適応能力がある，③現在の分業化したシステム開発の実際になじみやすいように，プログラムの設計仕様を①個別のプログラムを設計する仕様，②ファイルのデータフォーマット等プログラム間で共通な仕様，③入出力処理等の特殊な処理を記述する仕様，の3種類に分けて記述できる．記述実験により大規模バンキングシステムのオンライン業務プログラムの80%に適用可能であることが確かめられた．また手書き COBOL プログラムに比べ記述量が 1/3，目的プログラムのデバグ段階での発生バグ数が 1/4 に減少した．以上から SL の実用上の有効性が確認できた．

1. はじめに

プログラムジェネレータは①ジェネレータの入力自体にドキュメント性がある，②記述量が少ない，③処理系による記述内容の検証能力を充実できる，から飛躍的な生産性向上効果が期待できる．しかし従来のジェネレータは，データベース環境等も含めて特定利用向けの専用性が強く融通性に乏しい，分業化した記述への配慮に乏しいなどの問題があり多大な工数を必要とする大規模プログラムの開発には適用がままならないのが現状である．専用ジェネレータを利用目的別に開発することはジェネレータの開発費用も含めたトータルの生産性を考えれば得策ではなく適用性の広いジェネレータの開発が現実的である．そこで我々は以下の特徴を持つ設計用言語 SL の開発を進めた．

（1） 適用性向上のための特徴

（A） 設計仕様の記述に良構造プログラムを設計することを目的としたジャクソン法[1]（以下，JSP と言う）に代表されるデータ構造設計法を基盤にした仕様記述法を用いているため特定問題をモデル化したジェネレータに比べ汎用性がある．

（B） JSP の言うデータ構造は順次アクセスファイルを前提にして処理のビューで眺めた入出力データの論理的アクセスパスをもとにモデル化するものであるが，大規模プログラムで利用されている木構造型 CODASYL タイプデータベースのようにデータが階層化されかつ利用者がその構造を定義するデータでは，物理的アクセスパスが変わり得るため，JSP の言うデータ構造だけの情報では正しいプログラムを設計できない．そこで SL ではデータ構造の仕様として，処理のビューで眺めたデータ構造モデル（論理データ構造と言う）の仕様とともに物理的なアクセスパスのビューで眺めたデータ構造モデル（物理データ構造と言う）の仕様を記述させ，これらの仕様から正しいプログラムを生成できるようにした[2),3)]．

（C） SL が生成する目的プログラムの設計のうち，入出力処理等のプログラムの走行環境に依存する機能，問題向き関数等の専用性の高い特殊な処理については SL で定める標準中間言語と対応付けて利用者が設計できる仕組み[4),5)]を設けている．

（2） 分業化した記述のための特徴

現在の分業化したシステム開発の実際になじみやすいように，プログラムの設計仕様を①個別のプログラムを設計する仕様，②ファイルのデータフォーマット等プログラム間で共通な仕様，③入出力処理等の特殊な処理を記述する仕様，の3種類に分けて記述できる．

上記(1)の(A)のうち，JSP で言うデータ構造のみ

† Development and Evaluation of Program Design Specification Language SL by MITSURU INADA, TSUTOMU OKAMOTO, SATOSHI WATANABE and YUZO NAKAMURA (Software Production Technology Laboratories, Nippon Telegraph and Telephone Corporation).

†† NTT ソフトウェア生産技術研究所

を入力としてプログラムの自動生成を試みた類似の例として MODEL[6] が報告されているが，木構造データベースを処理対象としたプログラムの生成については含まれていない．また(2)，(3)の特徴を併せ持つ例はまだない．本報告では SL の概要，特徴，評価について述べる．

2. 設計法の背景と自動化範囲

2.1 データ構造設計法を選んだ背景

プログラム設計仕様の記述の考え方をデータ構造設計法を基盤にした理由を以下に述べる．

(1) プログラムを生成する仕様記述言語としては仕様記述法の形式性が高いほど効果的である．言語処理の仕様記述に用いられている句構造文法は，文字列をデータの単位とした入力データ構造と出力データ構造との間の写像を規定する方法で，形式性が高く機械処理のしやすい仕様記述法として知られている．一方，事務処理分野の仕様記述についても，データの単位をファイルのレコード等としてプログラムの入出力データ構造とその間の写像を規定する句構造文法流の考え方をとれば同等の形式性が得られ機械処理効果の向上が期待できる．またこの考え方は，事務処理といえども特定問題，例えば在庫管理とか給与計算といった問題を直接的に設計するためのものではなく汎用性のある設計概念である．

(2) 大規模プログラムで利用されているデータベースのデータ構造とデータ構造設計法には整合性があるが，さらにこの延長で機能強化や簡便にすることの可能性を持つ．

(3) データ構造設計法は手書きプログラムの良構造化のための設計法であり人手で作成したプログラムと比べて遜色のない能率の目的プログラムを生成する処理系の実現が期待できる．

2.2 設計手順と自動化範囲

本節では JSP を例にとってその設計手順と生産物の関係及び SL による自動化範囲についての概要を述べる．JSP では4段階の設計ステップと各ステップの生産物を規定している（**表 1**）．SL はデータ構造設計ステップの生産物であるデータ構造と，JSP の言う operations の列挙をするステップの生産物である operations のリスト（演算リストと言う）を仕様書として記述させ，プログラム構造への変換ステップと演算リストのプログラム構造への割り付けステップを自動化し COBOL プログラムを自動生成する．SL で記述する仕様書を句構造文法の流儀で説明すれば以下のようになる．プログラムの入出力データ構造の規定と，入力構文範疇と出力構文範疇の意味関数で意味を規定する．このとき構文範疇はデータ構造に対応する木（以下，ジャクソン木と言う）の節または葉に対応し，意味関数は演算リストに対応する．意味関数を SL では single assignment property を持つ代入文等によって記述する．

表 1 データ構造設計法（ジャクソン法）の概要
Table 1 Summary of data structure design method and coverage of automation by SL.

設計ステップ	生産物	SL
(1) データ構造の設計	データ構造	
(2) プログラム構造設計	プログラム構造（フローチャートまたは疑似コーディングによる概略制御フロー）	○
(3) 演算リストの設計	データ項目への代入文，チェック文，入出力命令文等の演算リスト	
(4) プログラム構造への演算リストの割り付け	プログラム（COBOL プログラム等）	○

（備考） ○：SL では自動化

3. 仕様書体系と処理系の特徴

SL の各仕様書と SL 処理系の役割を以下に述べる（**図 1**）．

3.1 仕様書記述の考え方

SL の仕様書は，大規模プログラムに適用したときの設計，記述の分業化，業務記述での詳細隠蔽，をねらいとして大きく三つの仕様に分けて記述する構成とした．以下に，各仕様書の役割と関係について概要を述べる．なお，SL のデータ構造の考え方，特徴については4章で詳しく述べる．

(1) プログラム仕様書

個別のプログラムを記述する仕様書であり最も詳細隠蔽レベルの高い仕様書である．電文，元帳等を処理のビューでモデル化した論理データ構造と元帳のデータ項目のチェック処理や更新処理等の演算リストを非手続き的に記述する．

(2) 共通仕様書

プログラム仕様書間で共通な仕様書であり通常はプログラム仕様書上からは隠蔽してもよい仕様書群であり総称して共通仕様書と言う．論理データ構造を正しく設計する上で必要な物理データ構造，データフォーマットの仕様，数え上げ型の定義等を記述する．

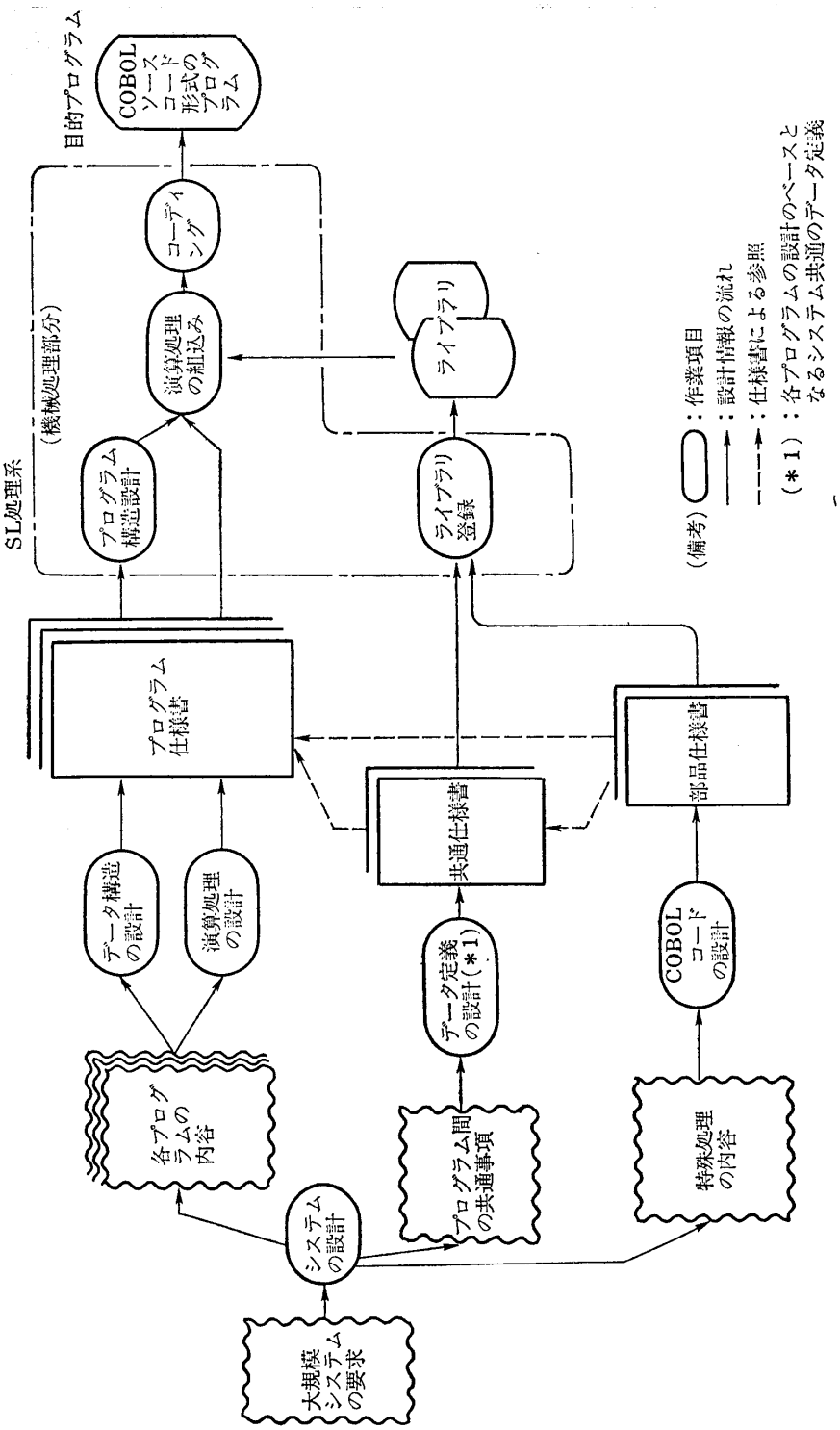

図 1 SL の仕様書と処理系の役割
Fig. 1 Role of specifications and SL processor.

(3) 部品仕様書

データ構造のアクセス処理等の走行環境との整合機能(いわゆる DB/DC 機能)や利息計算関数等の問題向き機能に対応する目的プログラムのコードは適用システムごとに微妙に異なることがある．このため，これらの特殊処理に対する目的プログラムのコード設計を処理系で固定化すると適用性の狭い処理系となる．

そこで，プログラム仕様書にはこれらの機能のための演算リストの詳細は書かせず，プログラム仕様書のコンパイル過程の標準中間言語と対応付けて利用者が COBOL コードを直接設計する仕組みを設け，利用者の環境に適合した目的プログラムが得られるようにするとともにプログラム仕様書の環境依存性を排除した．この仕組みに基づく仕様書を部品仕様書（単に部品とも言う）と言う．プログラム仕様書では部品名や関数名の参照のみを行う．

3.2 仕様書の種類と処理系の概要

3.1 節で述べた考え方に基づいて**表 2** に示す記述目的別に分類した仕様書を設けた．SL 処理系[5),7),10)]はこれらの仕様書を入力し，共通仕様書と部品仕様書を保存するライブラリへの登録とプログラム仕様書から目的プログラムを生成する機能を持つ．プログラム仕様書のコンパイル時には共通仕様書のライブラリ参照を行いプログラム仕様書の文法チェックやコンパイルに必要な情報を補うとともに部品や関数の参照に対しては，いったん部品名や関数名の情報を含む中間言語を生成しておき，コード生成処理でライブラリ化された部品仕様書から対応するコードを参照して目的プログラム中に展開し，完結した目的プログラムを得る．共通仕様書に対しても登録時に内容妥当性の検証を行うため共通仕様書は単なるマクロ機能とは異なる．共通仕様書に記述する内容は小規模利用のときのためにプログラム仕様書に直接記述することもできる．

表 2 仕様書の種類
Table 2 Specification system.

仕様書名	記述内容
プログラム仕様書	プログラムの論理データ構造，チェック，代入等の演算リストを主に記述する．一つのプログラム仕様書から一つの COBOL プログラムが生成される．
共通仕様書	プログラム中で規定する情報の中でプログラム間で一元的に管理すべきものをまとめた仕様書の総称．
データ仕様書	物理データ構造，フォーマットを記述する．データの使用目的に分けて複数の仕様書を作成することができる．
データ属性仕様書	数え上げ型を定義する．
インタフェース仕様書	サブルーチンコールのパラメータ，生成プログラムの名前を記述する．呼び出し側，被呼び出し側とも同一の仕様書が参照できインタフェースミスを防止できる．
部品仕様書	SL のコード生成部を利用システムへ適合させるための仕様書．入出力処理，リカバリ対策等の特殊処理，問題向き関数等のための演算リストを目的プログラム言語で記述する．

4. データ構造と演算リストの記述機能

SL の設計ではデータ構造設計法を基にして大規模バンキングシステムのアプリケーションプログラムの記述実験からの反映，処理系の実現性検証を進め，設計法及びその生産物を記述するための言語仕様[9)]を発展させた．以下に SL を用いたデータ構造の記述，演算リストの記述についての主要機能の概要と特徴，各仕様書との関係について述べる．

4.1 データ構造記述機能

データ構造の表現は，PL/1 の構造体に類似したテキスト形式で行う．その表現法の一覧を**表 3** に示す．以下に特徴的な事項を述べる．

4.1.1 データ構造表現能力の向上

(1) 表記法の改善

データ構造の表現法は JSP のジャクソン木の記法が著名であり SL の表現法(等価なテキスト形式)として利用できるかを考察した．ジャクソン木の構成はそのデータの基本単位となる elementary components (基本要素と言う) とその構造を表す列，繰り返し，選択からなる三つの composite types (構造と言う) から構成される．ジャクソン木の表記上の問題点としては基本要素に構造記号を付与する形をとっているため，ジャクソン木の節，葉ごとにどのような演算リストを設計すればよいかが分かりにくい．また，ジャクソン木の根は基本要素ではないのに基本要素と同じ表現になっておりその意味が曖昧である．SL では①基本要素に対する演算リストの設計ではデータに対するチェック，代入処理等を設計する，②構造に対する演算リストの設計では繰り返しや選択の条件等，構造固有の設計をする，として節，葉を見ながらそれぞれに固有の設計ができるように基本要素には構造記号を付与せず直上位の節（構造）に記述することにして木構造上の節，葉の役割が直感的に分かるようにした．

(2) 記述能力の向上

(A) 基本要素の階層

ジャクソン木は節に基本要素を持つ表現ができない

表 3 データ構造表現記号
Table 3 Data structure notation used in SL.

種別	構造					基本要素	
	列	繰り返し	選択	同列	プロセス		階層
記号	。(句点)	＊	！	＋	.(ピリオド)	△(空白)	——
SL記述例	００。A, ０１△B, ０１△C;	００＊A, ０１△B;	００！A, ０１△B, ０１△C;	００＋A, ０１△B, ０１△C;	００．A, ０１△B, ０１△C;	——	００△A, ０１＊B, ０２△C;
意味	AはB，Cの記述順の並び	AはBの繰り返し	AはB，Cのいずれか一つ	AはB，Cの順序のない並び	AはB，Cの記述順の並び（同一データの再使用等を表す場合）	基本要素を表す	基本要素に階層があることを表す

が木構造データベースのデータ構造は節にも基本要素を持つことのできる基本要素の階層化が行われている．このため基本要素を節にも記述できるようにした．基本要素の階層はアクセスパスを表し，下位の基本要素をアクセスするには上位の基本要素のアクセスが必要であることを表す．

（B） 同列構造

乱アクセスの更新ファイルを実現手段とする一つのデータ構造上の複数の基本要素を同時に入力し，相互にデータを参照する編集をした後，同時に更新をしたいことがある．JSP の構造の列ではこの処理について特に触れられていないが，基本要素ごとに入力，編集，更新の順序とするのが妥当な解釈と考えた．そこで，列とは別にこのようなデータ構造の記述ができる同列と呼ぶ構造を設けた．

（C） プロセス構造

いったん，出力データ構造として出力したデータを入力データ構造に切り換えて読む等，物理的な１ファイルについての多段の処理を一つのプログラムで記述したいことがある．JSP のデータ構造ではこれをどのように表現すればよいか明確でない．そこで，これを記述可能にするためのプロセス構造という構造を設けた．例えば，プロセス構造の配下には，入力時のデータ構造，出力時のデータ構造を処理順に記述することができ上記の処理が一つのプログラムで記述できる．

4.1.2 データ構造記述の構成

SL のデータ構造記述で最も特徴的な論理データ構造，物理データ構造及び両者の対応付けを行う抽出文についてその背景，機能概要と特徴について述べる．

（1） 背　　景

ジャクソン木は，処理のビューで基本要素を分類し，そのアクセスパスを表した論理的なデータ構造である．一方，データ構造には，プログラムの処理とは無関係な物理的アクセスパスのビューで見た実現法としての入れ物のデータ構造（例えば順次アクセスファイルであれば，処理と無関係な物理的な基本要素（レコード）の並びだけのデータ構造）がある．ジャクソン木の設計では入れ物の構造が順次アクセスファイルのみを仮定して，その範囲で書ける部分のみをジャクソン木で書かせ，乱アクセスファイル等の入れ物を扱うときは演算リストの設計の問題として片付けている．一方，SL が適用対象としているプログラムでは木構造データベースに代表されるように物理的な基本要素が階層化され，かつキーによる乱アクセスのできるファイルを主体に扱っており，演算リストの設計の問題としてしまうとデータ構造設計自体の意味が薄くなる．そこで，SL ではデータ構造の設計情報として，ジャクソン木と同じ意味付けを持つ処理のビューでみたデータ構造（論理データ構造）に，物理的アクセスパスのビューで見たデータ構造（物理データ構造）及び両者の対応をとるための設計情報（この情報を記述する文を抽出文と言う）を加えて，データベースを含むデータの処理をデータ構造設計の対象にできるようにした．

（2） データ構造と抽出文の役割

（A） 論理データ構造

プログラム仕様書には論理データ構造を，入力，出力，更新等の用途に分類して記述する．用途の情報はプログラム生成上の情報になるとともに演算リスト上のデータの参照関係の誤りチェックに用いる．JSP では入力と出力のデータ構造上の繰り返しデータ間の対応をとって両者のデータと処理の同期関係（JSP で言う対応関係)を示す設計過程がある． SL では出力または更新用途のデータ構造の繰り返し構造に相手繰り

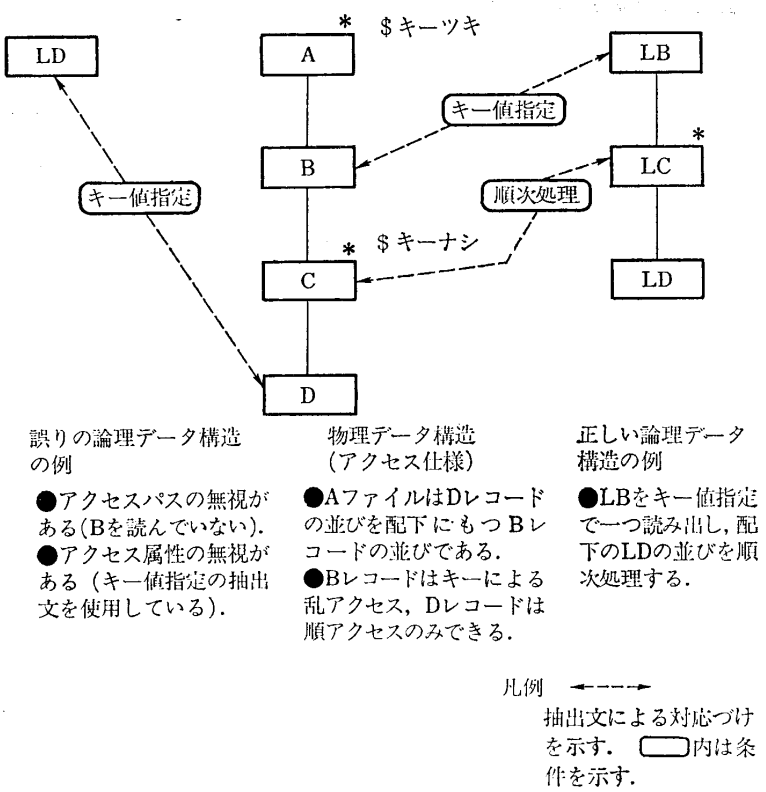

図 2 物理データ構造の役割
Fig. 2 Role of physical data structure.

返し構造名を指定することで対応関係を示す.

（B） 物理データ構造

物理データ構造はそのデータを使用するプログラムの設計者にどのような論理データ構造と抽出文の組み合せが設計し得るかを示す役割を持つ（図2）. 物理データ構造は論理データ構造と同様の記法で表現する. ただし，構造は列，繰り返し，同列のみ使用できる. 物理データ構造は物理的に可能なアクセスパスを表す. 例えば基本要素に階層があるとき下位の基本要素をアクセスするには上位のものからアクセスしなければならないこと（論理データ構造上でも上位のものからアクセスすることを書かなければならない）を示す. 物理データ構造に含まれる繰り返し構造にはその下位の基本要素が順次アクセスなのか，キーによる乱アクセスができるのか等を分類するアクセス属性と呼ぶ属性を指定する. アクセス属性には以下の3種類の属性があり一つを選択して指定する.

① $キーナシ……順次アクセスのみ可能.
② $キーツキ……キーによる乱アクセスのみ可能.
③ $キー……①，②の両者のアクセスの仕方が可能.

例えば物理データ構造上のある繰り返し構造に $キーナシのアクセス属性が指定されているとき，キー値指定の抽出文を書くと誤りとなる. 論理データ構造は処理のビューで設計するため同じ物理データ構造上のデータを処理するプログラムであってもプログラムが変わればその構造が変化する. 一方，物理データ構造は入れ物の仕様でありプログラム間に共通な仕様のため，そのアクセスの実現処理を記述する部品仕様書とともにシステム共通設計者が代表して設計し，データ仕様書としてライブラリ登録すると便利である.

（C） 抽 出 文

論理データ構造と物理データ構造の対応をとるための文であり論理データ構造の構造，基本要素ごとに指定する. 抽出文は上記の対応をとるほか，①繰り返しの終了条件，②乱アクセス時のキー値等のアクセス条件，③使用フォーマット，等を指定する役割を持つ. 抽出文にはこのほか，スタック処理，コントロールブレーク等の処理を指定するためのものがある. 抽出文は論理データ構造とともにプログラム仕様書に記述する.

4.2 演算リストの記述機能

JSP では演算リストについて，①データ構造で表したデータに対する業務記述のための演算リスト（代入文やチェック文等），②データ構造等の実現処理（入出力処理，問題向き関数の処理等）のための演算リスト，とを区別せず単一の演算リストを作成する. SL では演算リストを①はプログラム仕様書に，②は部品仕様書に，各々分けて記述する. 以下に演算リストの記述方式について述べる.

4.2.1 プログラム仕様書の演算リスト

個別のプログラムを記述する仕様書の演算リストであり，この記述の高度化を図ることは生産性向上に大きく寄与する. 以下に特徴を述べる.

（1） 抽出文や一時計算項目の定義文，代入文，チェック文等の細々とした文を，対象となる論理データ構造の構造及び基本要素ごとに分けて整理記述する記述の枠組みを設け，均質な記述をガイドしている.

（2） 代入文は one assignment property を持つ非手続き的記述とした. これによりプログラム仕様書の代入文は代入ではなく表明であるが利用者への親し

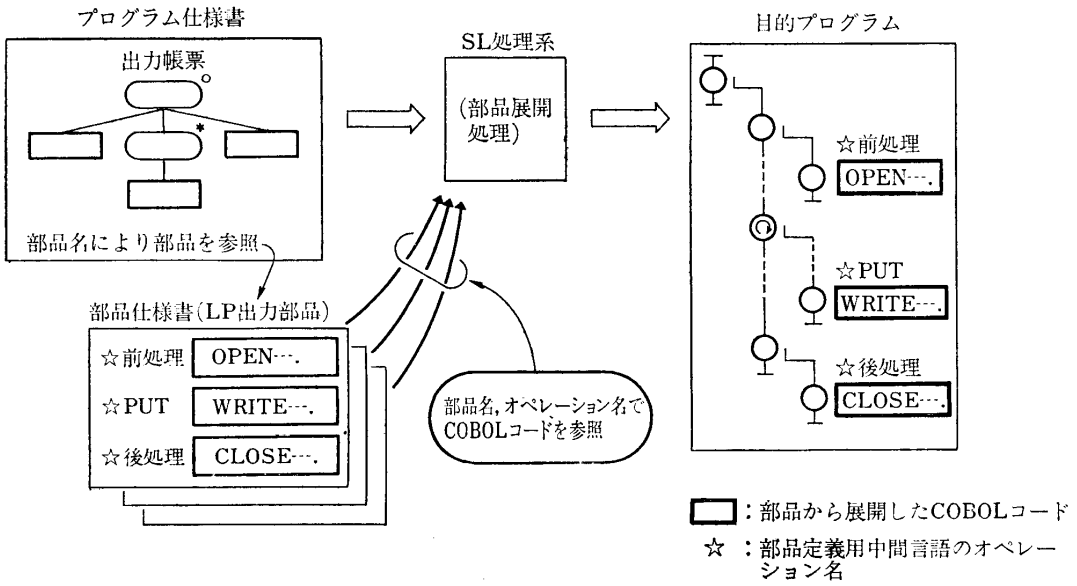

図 3 部品の展開
Fig. 3 Illustration of part expansion.

みやすさを配慮し，代入という表現を使用している．以下に特徴を示す．

（A） 代入式と代入条件を対にした代入文形式を設けた．一つの代入文には複数の対が記述できる．

（B） 代入条件はいくつかの代入文で共通になることが多く，このときの記述量の削減，実行時性能の向上のため多分岐条件文（ケース文）を設けた．代入条件の評価順序は(A)項の対のときも含めて実行時性能を重視して記述順とし，最初に満足したときの代入式の値を代入対象データ項目の値とした．

（C） 更新データ項目は旧値，新値の参照ができ，経験的な出現頻度を考慮して新値の参照のときにデータ項目名に修飾語（シン）を付けることにした．

（D） 部品仕様書で作成した利用者定義の関数を代入式等で引用できる．

（E） データ構造のアクセス結果等は組み込み関数を通じて問い合わせることができる．

4.2.2 部品仕様書の演算リスト

部品仕様書の役割を**図 3** に示す．部品仕様書は，データ構造等の実現処理である入出力等の演算リストを COBOL 言語で直接記述するための仕様書である．

COBOL プログラムの記述では，ある目的を達成するためには手続き部の実行文のセットばかりでなく関連する環境部，データ部での宣言文が必要なことが多い．例えばラインプリンタへ出力するために必要な COBOL コードは COBOL の標準入出力文を利用のとき手続き部の OPEN 文，WRITE 文，CLOSE 文のほか，環境部の FC 句，データ部の FD 句が必要である．部品仕様書ではこれらの一連の実行文，宣言文を SL で定める部品定義用中間言語のオペレーション名（例えば，入出力の実行文用のものとして前処理，後処理，PUT，宣言文用として FC，FD 等がある）と対応付けて一つの仕様書に一括して記述する．部品定義用中間言語は部品を参照するプログラム仕様書のコンパイル過程で処理系が生成するものと対応しており，最終的には生成中間言語中に含まれる部品名とオペレーション名をキーにしてライブラリを検索し，対応する COBOL コードに変換して目的プログラム中に分散展開する．プログラム仕様書からの参照は入出力用部品のとき部品仕様書名の参照のみでよ

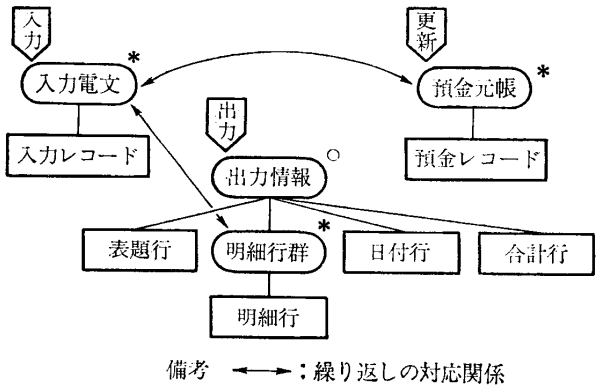

図 4 例題 (SL の記法と等価なデータ構造図による)
Fig. 4 Data structure of example problem.

```
1   1 ﾌﾟﾛｸﾞﾗﾑｼｮｳｼｮ(預金処理);
2   2 ﾛﾝﾘｷﾃｲ;
3    2.1 ﾛﾝﾘﾃﾞｰﾀｺｳｿﾞｳｷﾃｲ;
4    〈ﾆｭｳﾘｮｸ〉:
5     00* 入力電文,
6      01 入力レコード;
7    〈ｺｳｼﾝ〉:
8     00* 預金元帳=入力電文,
9      01 預金レコード;
10   〈ｼｭﾂﾘｮｸ〉:
11    00. 出力情報,
12     01 表題行,
13     01* 明細行群=入力電文,
14      02 明細行,
15     01 日付行,
16     01 合計行;
17  2.2 ｼｮｳｻｲｼｮﾘｷﾃｲ;
18   〔入力電文〕:
19   〈ｾﾝﾃｲ〉:
20    〈1〉 @<=ＩＮＰＵＴ電文;
21   〈ﾎｿﾞｸ〉:
22    00 入金件数計 $DEC(2) $ｼｮｷﾁ(0)
23         :=ｾﾝ.@+1:入力レコード.入出金区分="入金;
24    00 出金件数計 $DEC(2) $ｼｮｷﾁ(0)
25         :=ｾﾝ.@+1:入力レコード.入出金区分="出金;
26    00 入金金額計 $DEC(10) $ｼｮｷﾁ(0)
27         :=ｾﾝ.@+入力レコード.金額:入力レコード.
28              入出金区分="入金;
29    00 出金金額計 $DEC(10) $ｼｮｷﾁ(0)
30         :=ｾﾝ.@+入力レコード.金額:入力レコード.
31              入出金区分="出金;
32    00 西暦取引日 $DEC(6) $ｼｮｷﾁ(0)
33                :=ﾍﾝｶﾝ(入力レコード.取引日);
34   〔入力レコード〕:
35   〈ｾﾝﾃｲ〉:
36    〈1〉 @<=ＩＮＰＵＴレコード;
37   〈ﾁｪｯｸ〉:
38    〈1〉 金額 > 99999999 ? %ｴﾗｰ-(101);
39   〔預金元帳〕:
40   〈ｾﾝﾃｲ〉:
41    〈1〉 @<=元帳ファイル;
42   〔預金レコード〕:
43   〈ｾﾝﾃｲ〉:
44    〈1〉 @<=ｾﾚｸﾄ(元帳レコード) ｷｰ (入力レコード.
45                    口座番号);
46    〈1〉 ｺｳｼﾝ;
47   〈ﾎｿﾞｸ〉:
48    00 計算値 $DEC(10) $ｼｮｷﾁ(0)
49       :=残高-入力レコード.金額:入力レコード.
50            入出金区分="出金;
51   〈ﾁｪｯｸ〉:
52    〈1〉 計算値 < 0 ? %ｴﾗｰ (102);
53   〈ﾖｳｿｾｯﾃｲ〉:
54    〈1〉 残高:=@+入力レコード.金額:入力レコード.
55                 入出金区分="入金
56              :=@-入力レコード.金額:入力レコード.
57                 入出金区分="出金;
58    〈1〉 取引件数:=@+1;
59     〔出力情報〕:
       ⋮
94    2.3 ﾚｲｶﾞｲｼｮﾘｷﾃｲ;
95     〔ｴﾗｰ〕:
96      〈ﾊﾟﾗﾒｰﾀ〉;
97       00 ｴﾗｰｺｰﾄﾞ  $DEC(3);
98      〈ﾚｲｶﾞｲｼｮﾘ〉;
99       〈1〉 ｲｼﾞｮｳｼｭｳﾘｮｳ;
100 3 ﾌﾞﾂﾘｷﾃｲ;
101  3.1 ﾌﾟﾛｸﾞﾗﾑｷﾃｲ;
102    ﾌﾟﾛｸﾞﾗﾑﾒｲ=YOKIN;
103    ﾌﾟﾛｸﾞﾗﾑｹｲｼｷ=ﾒｲﾝ;
104  3.2 ﾃﾞｰﾀｷﾃｲ;
105   〔ＩＮＰＵＴ電文〕;
106   〈ﾌﾞﾂﾘﾃﾞｰﾀｺｳｿﾞｳ〉;
107    00* ＩＮＰＵＴ電文  $ｷｰﾅｼ,
108     01 ＩＮＰＵＴレコード;
109   〈ﾌｫｰﾏｯﾄ〉;
110    00 ＩＮＰＵＴレコード,
111     01 取引日     $DEC(6),
112     01 口座番号   $DEC(6),
113     01 入出金区分 $区分コード,
114     01 金額       $DEC(10),
115     01 領域       $CHR(113);
116   〈ﾒｶﾆｽﾞﾑ〉;
117    @DATAIN=();
118   〔元帳ファイル〕;
119   〈ﾌﾞﾂﾘﾃﾞｰﾀｺｳｿﾞｳ〉;
120    00* 元帳ファイル  $ｷｰﾂｷ,
121     01 元帳レコード;
122   〈ﾌｫｰﾏｯﾄ〉;
123    00 元帳レコード,
124     01 口座番号   $DEC(6) $ﾎｽﾄﾒｲ(KOBAN),
125     01 残高       $DEC(10),
126     01 取引件数   $DEC(2);
127   〈ﾒｶﾆｽﾞﾑ〉;
128    @UPDATEK=();
129   〔出力ファイル〕;
130   〈ﾌﾞﾂﾘﾃﾞｰﾀｺｳｿﾞｳ〉;
131    00* 出力ファイル  $ｷｰﾅｼ,
132     01 出力レコード;
133   〈ﾌｫｰﾏｯﾄ〉;
      ⋮
159   〈ﾒｶﾆｽﾞﾑ〉;
160    @DATAOUT=();
161  3.3 ﾃﾞｰﾀｿｰｸｾｲｷﾃｲ;
162   〔区分コード〕;
163   〈ﾛﾝﾘｷﾃｲ〉;
164    00 区分コード=〔"入金,"出金〕;
165   〈ﾌﾞﾂﾘｷﾃｲ〉;
166    00 区分コード< $区分コード:$DEC(1)> =〔1,2〕;
167 4 ｼｮｳｼｮｵﾜﾘ;
```

図 5 プログラム仕様書の記述例
Fig. 5 Illustration of program specification written in SL.

```
 1    1 ﾌﾞﾋﾝｼｮｳｼｮ(DATAOUT);
 2    2 IOﾌﾞﾋﾝｷﾃｲ;
 3     2.1 ｲﾝﾀﾌｪｰｽｷﾃｲ;
 :
12      2.2 ﾃﾝｶｲｼｮｳｷﾃｲ;
13      <<ﾏｴｼｮﾘ>>;
14              OPEN   OUTPUT   C-FILE.
15      <<PUT>>;
16              WRITE   @ﾎｽﾄﾒｲ(@OUTR).
17      <<ｱﾄｼｮﾘ>>;
18              CLOSE   C-FILE.
19      <<END>>;
20    3 ｷｮｳﾂｳﾃﾝｶｲｷﾃｲ;
21     3.1 ｾﾝｹﾞﾝｷﾃｲ;
22      <<FC>>;
23              SELECT C-FILE ASSIGN TO SYSOUT;
24              FILE   STATUS   @GEN(1).
25      <<FD>>;
26          FD   C-FILE LABEL RECORD IS STANDARD.
 :
34      <<END>>;
35    4 ｼｮｳｼｮｵﾜﾘ;
```

図 6 部品仕様書の記述例
Fig. 6 Illustration of I/O part specification written in SL.

く，オペレーション名は処理系が自動的に参照する．

5. 記述例と記述則の概要

記述例として**図4**に示す簡単な元帳更新プログラムについて，プログラム仕様書の記述例を**図5**に，プログラム仕様書で参照している出力用 DATAOUT 部品の部品仕様書の記述例を**図6**に示す．SL の全仕様書は章節段落構成をとっている．またキーワード，名標は，英数カナ文字を基本としているがプログラム仕様書，共通仕様書にのみ名標に漢字を使用できる．データ構造の記述は PL/1 の構造体と類似の形式であり，構造記号はレベル番号の右側に示す．

（1） プログラム仕様書

プログラム仕様書は四つの章から構成される．2章には論理データ構造と演算リストを記述する．2.1 節で論理データ構造を入力，出力等の用途に分けて記述する．繰り返しの対応関係は出力，更新用途の繰り返し構造に相手の構造名を記述する．2.2 節には論理データ構造の構造，基本要素ごとに見出しを付けて演算リストを記述する．各構造，基本要素ごとに①抽出文等を記述する前提段落，②一時計算項目を記述する補足段落，③基本要素のデータをチェックするチェック段落，④基本要素が出力または更新の用途のとき代入文を記述する要素設定段落，を記述する．要素設定段落では自己の基本要素にのみ代入できる．2.3 節には例外処理の演算リストを記述する．2.2 節のチェック段落のチェック文等から例外処理への制御の移行を記述できる．3章では物理データ構造やフォーマット，入出力部品等の部品参照，数え上げ型の定義[8]等を記述する．3章については共通仕様書を利用すれば共通仕様書の参照文だけとなる．

（2） 部品仕様書

部品仕様書は四つの章から構成される．1章では部品名を定義する．プログラム仕様書で入出力用部品を参照するとき，物理データ構造と対応付けてこの部品名で参照する．目的プログラムのコードは部品定義用中間言語のオペレーション名（11種類ある）と対応付けて記述する．2章には手続き部に展開する内容を，3章には宣言部，データ部に展開する内容を記述する．コード中には部品を汎用化するための COBOL プログラムとしての一意名標取得関数（@GEN），プログラム仕様書の漢字名標を英字名標に変換した名標を取得する関数(@ホストメイ)等の展開時に実行される関数が記述できる．

6. 評　　価

（1） 記 述 能 力

大規模バンキングシステムのオンライン業務プログ

ラムの典型パターンを含む10本を抽出した机上記述実験の結果，２本にデータ構造記述能力上の問題があった．２本が適用できなかったのは①順次アクセスファイル上で逆順読みの処理を表現できない，②入力データに繰り返しがなく，入力と対応関係のとれない出力データのみの繰り返しが表現できない，ことによる．

また，JSP の言うストラクチャクラッシュは SL のプログラム仕様書上では対処できない．ただし，境界不一致問題は境界調整をする入出力用部品（JSP の言うinverted routine 相当の処理を記述)を作成すればプログラム仕様書上では解決する．

部品仕様書の記述能力ではデータベースを含めた入出力について部品化できることが分かった．部品は問題向き関数，特殊処理用部品も含めて約100部品，業務プログラムとの規模比で数%程度の部品を作成すればよいとの見通しを得た．

（2） 記述量比較

手書き COBOL プログラムの記述行数と比べた SL プログラム仕様書の記述行数は部品による展開も含めて約３分の１であった．この値は専用ジェネレータに比べると高くはないが SL は①適用性を広くすることをねらいとした，②記述量を徹底して少なくするには仕様書入力の支援系を作ればよいとした，ことによる限界である．特に記述のガイドのための枠組み記述が多いため小さなプログラムでは見掛け上の記述量が多くなる．データ構造部分からの展開量は余り多くはないが制御構造，制御用ワークデータの生成，アクセス結果の判定処理等の自動生成コードがあるほか，複数のデータ構造から一つのプログラム構造に矛盾なく変換するための設計工数の削減効果で生産性向上に寄与する．プログラム仕様書の演算リストでは条件付き代入文，関数参照等からの展開が展開率の向上に効く．

（3） 目的プログラムの信頼性

手書き COBOL プログラムに比べ，テスト工程で検出されたバグ数は約４分の１であった．コンパイル中に取れずにテスト段階まで残存したバグの多くは要求仕様からのマッピングミス，例えば条件判定の反転等であった．

（4） 性　　能

NTT の DIPS 上の COBOL コンパイラとの翻訳時間比，手書き COBOL プログラムと目的プログラムとの実行時性能比較についての評価を実施した．翻訳時間は主として非手続き的記述の仕様から手続きへ変換するためのデータフロー解析処理の負荷が高いため約２倍となった．実行時性能は手書き COBOL プログラムに比べステップ数，メモリ量とも約 20-30% 増であった．試作処理系のため，共通条件判定の括り出し等の最適化，入出力バッファの割り付け方の最適化の余地が残されている．

7. むすび

適用性の向上及び大規模プログラムの開発に効果的に利用できることをねらいとした SL の概要及び特徴について述べた．バンキングシステムのオンライン業務プログラムの記述実験では約 80% のプログラムの記述能力のあることが判明した．また手書きプログラムに比べ記述行数比較で 1/3 に，テスト工程での発生バグ数が約 1/4 に減少した．適用性向上のための SL の特徴は①汎用性のあるプログラム設計法に基づく仕様記述をしていること，②データ構造の設計に論理データ構造，物理データ構造，抽出文の概念を取り入れたことによりメモリ上のデータから木構造データベースを含む入出力データをデータ構造設計の対象として統一的に扱えるようにしたこと，③入出力等の特殊処理の目的プログラムの設計を利用者が行う部品化方式を導入し多様な環境に適合できるようにしたこと，である．

プログラム仕様書，共通仕様書，部品仕様書の三つに分けた仕様記述方式は各仕様書間の独立性があるとともに詳細隠蔽効果があり，①大規模プログラムの記述の分業化を行うことができる，②レビューがしやすく誤りの作り込み防止ができる，③ SL 記述の仕様書の再利用を促進する，の効果が期待できる．仕様書の均質記述をねらいとした章節段落構成及び論理データ構造の構造，基本要素に着目した演算リストの記述の枠組みの考え方は，VDT 利用の仕様書入力の支援系を設計する上で整合性があり，入力時のガイド，記述（キータッチ）量の削減，入力時の簡易文法チェック等を実現すればさらに SL を効果的なものにすることができると考えている．現在，SL の試作処理系を NTT のDIPS 上でインプリメントした段階であり，今後各種大規模リアルタイムシステム等の業務プログラムへの適用を進めさらに実用上の有効性を確認する予定である．

謝辞 本研究をご指導，ご支援いただいた花田收悦ソフトウェア生産技術研究所長，細谷僚一プログラム言語研究室長，伊東洋一生産システム研究室長，SL

関係各位へ厚くお礼申しあげる.

参考文献

1) Jackson, M. A.: *Principles of Program Design*, Academic Press, New York (1975).
2) 岡本ほか：設計言語におけるデータ構造の階層的表現について，58. 10，情処学会第 26 回全国大会，pp. 457-458 (1983).
3) 稲田ほか：データ中心型設計言語を用いた効果的設計方法について，59. 3，電子通信学会総合全国大会，p. 1721 (1984).
4) 渡辺ほか：設計言語における部品構成法の検討，59. 9，情処学会第 29 回全国大会，p. 563 (1984).
5) 岡本ほか：拡張可能な非手順型言語方式の一提案，58. 4，電子通信学会総合全国大会，p. 656 (1983).
6) Prywes, N. S.: *Automatic Generation of Computer Programs*, *Advances in Computers*, Vol. 16, pp. 57-125, Academic Press, New York (1977).
7) 岡本ほか：データ構造とオペレーションに基づくプログラム自動生成法の一提案，58. 10，情処学会第 26 回全国大会，pp. 503-504 (1983).
8) 中村ほか：設計言語におけるデータ属性の階層的定義について，57. 10，情処学会第 25 回全国大会，pp. 495-496 (1982).
9) 稲田ほか：設計用言語 (SL) の言語仕様，情処学会ソフトウェア工学研究会，42-3 (1985).
10) 渡辺ほか：設計用言語 (SL) の処理方式，情処学会ソフトウェア工学研究会，42-4 (1985).

(昭和 60 年 9 月 6 日受付)
(昭和 61 年 3 月 20 日採録)

稲田　満（正会員）

昭和 21 年生．昭和 44 年東京電機大学工学部電子工学科卒業．同年日本電信電話公社入社．現在ＮＴＴソフトウェア生産技術研究所勤務．主に言語処理プログラム，拡張型言語，開発支援技術の研究実用化に従事．電子通信学会会員．

岡本　務（正会員）

昭和 20 年生．昭和 45 年大阪大学大学院修士課程（電子工学専攻）修了．同年ＮＴＴ入社．現在，同ソフトウェア生産技術研究所勤務．オペレーティングシステム，データベース管理システム，ソフトウェア工学の研究実用化に従事．電子通信学会会員．

渡辺　敏（正会員）

昭和 25 年生．昭和 48 年広島大学工学部電気工学科卒業．同年日本電信電話公社入社．現在，ＮＴＴソフトウェア生産技術研究所勤務．これまで，言語処理系，設計/製造/テスト支援技術の研究実用化に従事．電子通信学会会員．

中村　雄三（正会員）

昭和 28 年生．昭和 51 年慶応義塾大学工学部計測工学科卒業．昭和 53 年同修士課程修了．同年日本電信電話公社入社．現在，ＮＴＴソフトウェア生産技術研究所勤務．主にプログラム言語，開発支援技術の研究実用化に従事．電子通信学会会員．