情報処理技術遺産

構造化プログラミング言語 SPL

～技術遺産と SPL の紹介～
～技術小史と思い出～
～技術編～

技術士(情報工学)
加藤木和夫
平成 31 年 1 月

はじめに

今から 40 年前に筆者も関わり開発された**ソフトウェア製品（構造化プログラミング言語 SPL）が 2018 年 3 月に情報処理学会の情報技術遺産に認定**されました。本小冊子では情報技術遺産と SPL の紹介、開発当時のソフトウェア技術小史と個人的な思い出について記述し、最後に SPL の技術的な課題はどのようなものであったについてコンパイラ技術の観点から纏めてみます。

１．情報処理技術遺産とは

情報処理技術遺産は情報処理学会が認定するもので、今までの 10 年間で 100 件が認定されました。内訳は 96 件がハードウェア製品他、4 件がソフトウェア製品です。情報処理学会なのでソフトウェア製品が多くなるはずですが、意外にソフトウェア製品は少ないです。これは遺産に認定される製品は社会に多大な貢献をした実績が必須であると同時に、遺産として形が残っていなければならないためです。ソフトウェアの場合は現物（設計書やプログラム）が揃っていることが遺産認定に必須となってきます。しかし、動作中あるいは過去に動作していたソフトウェア製品で初期の現物が保存されているものは意外とありません。なお、論文は遺産現物の対象ではありません。

SPL は技術レベルの高さ、社会貢献としては実に 2700 件のシステムに適用され、現物として設計書、プログラムが残っていました。（情報処理学会の記事から写真転載）

▲構造化プログラミング言語SPL コンパイラソースリスト　　（左）
▲構造化プログラミング言語SPL 開発用ドキュメントとマニュアル　（右）

なお、技術遺産認定事業は 2017 年度で終了ということなので、SPL は最後に認定された技術遺産ということになります。

参考までに今までに技術遺産となったソフトウェア製品 4 件は下記です。

・ワープロソフトの一太郎（ジャストシステム社）

・HARP5020　FORTRAN（日立）

・CASE ツール（NEC）

そして・構造化プログラミング言語 SPL（日立）

2．開発メンバーは

開発者は公的には法人の日立製作所ですが、最初のバージョンの開発者名は日立大みか事業所内にある「分散コンピュータ博物館」*に SPL の開発にかかわる解説パネルが展示されており、そこには次に示すメンバー名が記載されています。

体制は全体統括に中田育男氏（シ研：主任研究員）、プロジェクトリーダーに林氏が当たり、言語設計は林氏と野木氏、言語の評価は浜田氏、言語処理系（コンパイラ）の設計は基本設計が野木氏と中所氏を中心に篠本氏、森氏、高藤氏、加藤木で、皆このころ 20 代後半から 30 代前半の気鋭のソフトウェア研究者、技術者であった。

SPL開発プロジェクト体制（1975年当時）

特研75A21「計算機制御ソフトウェア一貫生産システム（SP）の開発」（1975/10-1977/9)
第4分科会　（ソフトウェア設計支援システム：主査（シ研）中田主任研究員）として開発

特研主任研究者　：（お）宅間ソ2設部長
特研副主任研究者：（お）高井副技師長、（シ研）中田主任研究員、（日研）平沢主任研究員

分担	担当	所属	職位
取纏め＆言語機能企画	林　利弘	（お）ソ2設ソ技グループ	技師
言語仕様設計・評価	野木兼六	（シ研）	研究員
	浜田亘曼	（日研）	研究員
処理系基本設計・評価	中所武司	（シ研）	研究員
	篠本　学	（お）ソ2設ソ技グループ	企画員
	高藤政雄	（日研）	企画員
	加藤木和夫	（HEC）	企画員
処理系詳細設計	篠本　学	（お）ソ2設ソ技グループ	企画員
	森清三	（お）ソ2設ソ技グループ	企画員
	加藤木和夫	（HEC）	企画員
	柳沢	（お）ソ2設ソ技グループ	企画員
処理系記述システム設計	野木兼六	（シ研）	研究員
処理系記述システム製作	赤堀（石田）厚子	（シ研）	企画員

初版開発（1975/3-1976/10）工数：200人月　　1978年社長技術賞特賞受賞

15

*技術遺産を展示する場所が必要なので、以前からあった「大みか制御史料室」が今回の受賞に合わせて同時に博物館として学会から認定されました。

３．開発経緯

「SPL の言語仕様」は昭和 49 年後半から 50 年の初めにかけて、言語研究者の野木氏と制御用に必要な言語仕様イメージを持った林氏との間で検討と設計がなされました。言語仕様を実現する「SPL コンパイラ」の基本設計はシステム開発研究所に関係者が集まり昭和 50 年 4 月から 8 月にかけて行われ、筆者も派遣されて参加しました。開発（実装）はその年の 9 月から日立大みか事業所で開始され約 2 年で最初のバージョンが完成しました。その後は数年にわたり言語仕様の変更とコンパイル速度の高速化が行われ、以後数十年にわたり機種別対応や C 言語対応などの改造作業が続きました。

筆者は当初の 7 年間の開発に全面的に携わりました。SPL コンパイラは新規オリジナルの難しいソフトウェアでありましたが、アプリケーション製品の大きな社外事故につながる不具合は皆無でした。しかし、アプリケーション製品を開発する途中での不具合は製品ごとに発生し、その対応には数年にわたり苦慮しました。不具合の大きな要因は制御用ソフトウェア開発向けに考え出された SPL 言語仕様からくる通常のコンパイラにはない難しい構造でした。詳細は「技術編」に記述します。

筆者は 7 年後の機種対応の新規バージョン開発時点で体調を崩し開発現場を離れることを余儀なくされました。

なお、20 年ほど前から SPL 初期バージョン開発メンバーが「中田先生を囲む会」の名前で年に一回食事会を開催しています。中田先生は情報処理学会のフェローで日本を代表する情報処理の研究者です。メンバーは中田先生、野木氏、林氏、中所氏、森氏、石田氏、加藤木の 7 名です。

４．認定記念報告会と認定盾の授与

2018 年 12 月 6 日に日立大みか事業所にて「技術遺産報告会」が行われました。出席者は情報処理学会から認定委員 1 名、大みか事業所から所長をはじめ幹部、招待者として開発メンバーの 9 名と当時のユーザ代表（久保氏）および社外教育担当者（中野氏）が出席しました。午前中に「大みかコンピュータ博物館」の見学を行い、昼食をはさみ午後に開発メンバーに対して技術遺産認定証の贈呈および講演会が実施されました。

贈呈式では情報処理学会の旭氏（歴史特別委員会 幹事）が技術遺産の説明、次いで若い技術者を講堂に集めて林氏が「トップダウン型構造化プログラミング言語 SPL の開発を振り返って」（60 分）という講演を行い、開発者 9 名の一人一人が思い出と若い人へのメッセージを伝える報告会が行われました（1 人数分）。筆者も実装・保守の思い出をスピーキングしました。

開発メンバー9 名のうち 6 名は中田先生を囲む会のメンバーで、年に一回食事会で顔を合わせているので初めから和やかな雰囲気でした。他の 3 人ともすぐに打ち解け、と

くに筆者は篠本氏とは 30 数年ぶりにお会いしましたが、すぐに 40 年前の開発のために戸塚での寮生活を共にした時間に戻り、最近の生活について話が弾み得難い時間を過ごすことができ良い 1 日でした。

左：情報処理学会からの認定証（記念の盾）右：日立製作所大みか事業所からの感謝状

認定証受賞者　中田育男氏、野木兼六氏、中所武司氏、浜田亘曼氏、高藤政雄氏、林利弘氏、篠本学氏、森清三氏、加藤木和夫
（他に　高井氏（故人）、宅間氏（病欠）、平沢氏（遠方））
※認定証は昭和 52 年情報処理学会全国大会論文発表者と連名者

次ページに日立大みか事業所のホームページを掲載します。

●日立製作所のお知らせ（HP）

2018 年 3 月 6 日

株式会社日立製作所

大みか事業所にて開発した「構造化プログラミング言語(SPL)」が
「情報処理技術遺産」に認定
同時に、大みか事業所の開発・製造製品を展示する「大みか制御史料室」が
「分散コンピュータ博物館」に認定

株式会社日立製作所(以下、日立)は、このたび、一般社団法人情報処理学会(会長: 西尾 章治郎／以下、情報処理学会)より、大みか事業所(茨城県日立市)にて開発した「構造化プログラミング言語」が「情報処理技術遺産」の認定を受けたことをお知らせします。同時に、大みか事業所が開発・製造に携わった製品を展示する施設「大みか制御史料室」が、情報処理学会より「分散コンピュータ博物館」として認定されました。

「情報処理技術遺産」および「分散コンピュータ博物館」は、わが国のコンピュータ技術発達史上、重要な技術や製品などを次世代に継承していく上で意義を持つ歴史的文物の保存と活用を図るため、情報処理学会が制定した制度です。

今回の認定は、1970 年代のコンピュータ制御システムの草分け時代を支えたプログラミング言語の先進性と、先人の成果を実物とともに展示し、後世に伝えようとする取り組みが評価されたものです。日立は今後も、新しい情報処理技術の開発と製品提供を通して、コンピュータを利用するさまざまな産業の発展に貢献していきます。

「情報処理技術遺産」：構造化プログラミング言語（SPL）

「構造化プログラミング言語(SPL)」は、大規模化が進む制御システムにおいてソフトウェア開発の生産性向上をはかるため、新世代高級言語として、1976 年に大みか事業所で開発されました。トップダウン（階層）型の構造化プログラミングを可能とし、各階層では抽象化された概念を自然言語風に記述できるなど、当時から先進的な技術、機能を装備しており、社会インフラを支える制御システムの構築に向け、ソフトウェア開発の生産効率を向上させるとともに高品質化へ大きく寄与しました。本言語は、1976 年から 2016 年までにわたり、鉄鋼制御システムや列車運行管理システムなど、国内外約 2,700 のさまざまな制御システムに適用されました。

技術小史と思い出編

１．コンピュータ小史

筆者がソフトウェアの開発に関わり始めた 1970 年代初頭は、入出力媒体が紙テープ、カードや M/T であり、CPU の動作がランプの点滅で確認できる時代でした。コンピュータは汎用（事務計算、科学技術計算）、制御用にかかわらず大変貴重な資源であり、自由に使うことはできず、使うには計算センターで時間を割り当てられ真夜中にテストなどをしました。その後、コンピュータの発展は目覚ましく、制御用コンピュータでもオペレーティングシステム（OS）はメーカ独自のもの（PMS という名称でした）から UNIX をベースにするものへと時代と変わりました。

また、開発に使用するプログラミング言語もアセンブラ言語から FORTRAN（科学技術系）や COBOL（事務系）系を経て、C 言語や Java 言語へと変わり、周辺の支援ソフトウェアも充実し、その開発環境の進化はここ数十年の進歩はまさに隔世の感です。

日立の制御用の世界でもアセンブラ言語に始まり FORTRAN をベースにした工業用 FORTRAN（日立では PCL（Process Control Language））から今回受賞した制御プラントのデータ構造を反映した構造化プログラミング言語 SPL へと発展しました。その後は制御用ソフトウェア開発でも C 言語系が使われるようになりました。ただ、個人的には制御用ソフトウェアの信頼性の点では少し後退した感想を持っています。

一方、コンピュータの原理はノイマン型（ハードウェアの上にソフトウェアを動作させるタイプ）のままであり、また、ソフトウェアの開発環境は進歩してもソフトウェアが最終的には人の手により作り上げられるのは今のところ同じです（完全自動化には至っていません）。

大きく変化した部分と本質的には変わらなかった部分の両方を数十年で体験でき、その一端を一技術者として経験できたことを幸運だったと思います。

２．個人的なプログラミング小史

ソフトウェアの仕事にずっと携わることになるとは思っても見ませんでした。学生時代は物理学科に属していましたが、当時勃興しつつあったコンピュータとは無縁に過ごし、就職して初めてコンピュータにふれました。しかもコンピュータは一般的に知られた汎用コンピュータではなく、電力系統や新幹線運行などを制御する制御用コンピュータであり、プログラミング言語は機械語に近いアセンブラ言語でした。そのため、その分からないことといったら尋常でありませんでした。もとより参考書はなく、周りに聞

いて回るしか解決法がないという時代でしたが、その周りにいる人たちでさえ経験が乏しい時代でした。

研修で初めて作成したプログラムはアセンブラ言語でダンプルーチン（メモリの内容を印刷する）を呼び出す簡単なものでしたが、カードのシーケンスチェックのエラーがおびただしく出力され途方にくれました。毎日寝ると、アセンブラの命令語が出てくる悪夢に悩まされ、プログラム作りには向いていないのではと思う毎日でした。

入社した（1971 年）6 月ごろに研修も終わり、本来はデータベース管理システムの開発グループに配属の予定が、コンパイラ開発グループで人手が足りないということで、ほんのお手伝いでコンパイラグループに加わりました。これをお手本にと渡されたフローチャートは英文の FORTRAN コンパイラの一部で全く意味が分からず、お手上げの状態でした。

9 月ぐらいにやっと、アセンブラ言語で 200 行ぐらいのサブルーチンを作り上げ、次いで、コンパイラのオブジェクト生成フェーズといわれる部分を担当し、年も明けた 1 月には約 10,000 ステップ（命令数）のフェーズを完成することができました。その頃にはやっとコンパイラの意味が理解でき始めました。

データベース管理システムは開発中止となり、コンパイラグループとして残ることになり、それから 2 年ほどコンパイラの改良作業に追われて、毎日がバグ（不良）との戦いでした。

しかし、アセンブラ言語を用いて腕力でプログラムを作れるようになりましたが、新しい問題に出会ったときに、プログラムはどのように設計するのかというのはどうも分かりませんでした。そうこうするある日、偶然「ワーニェ」の本を手に取ったときの驚きは衝撃の一言でした。誰も教えてくれなかったプログラムの作り方がそこには書かれていました。

※『ワーニェ方式によるプログラミング学習（上／下）』（日本能率協会）

コンパイラの改良作業も 3 年ほど続くとさすがに飽きがきて、プログラム作りはもういいか、という心境になってきました。それを林利弘氏が見透かしたのか、新しいプログラミング言語のコンパイラを開発する話が持ちあがった時に（これが SPL です）、参加するようにとの指示が出ました。まず、コンパイラの開発のための汎用トランスレータシステム（UTS）の勉強のために日立のシステム開発研究所へ派遣されました。ところが、UTS の移植開発担当者が経験不足のために、まず SPL 開発の前に UTS の設計を担当することになりました。

※UTS 汎用計算機向けに開発されたコンパイラやプリプロセッサを記述するシステムを
制御用計算機向け（磁気ドラム方式）に移植・開発を行った。

UTS はインタプリタ型のトランスレータ記述言語を内蔵しており、その設計者は気鋭の研究者野木兼六氏でした。コンパイラの機種間移行はインタプリタを移行すればコンパイラも移行できるという素晴らしいアイデアでした。最終的にはインタプリタの速

度の問題で新機種上でのコンパイラの再製作となりましたが。

ところで、飽きてきていたプログラム開発に、この UTS の製作で突然面白さが戻ってきました。それはプログラムというよりソフトウェアの開発手段やツール設計の面白さでした。派遣されて 1 週間ぐらいした頃、野木氏へ「ハッシュとは何ですか」と質問したことがありました。答えは「え！知らないの。本当に君はコンパイラを 3 年も作っていたの。そんなの自分で勉強してよ。」と呆れられました（後年、野木氏にその話をすると、えー、そーだったの、とかで覚えていませんでした）。

そして、1 ヶ月たった頃、年末に飲みに行ったとき、「君ぐらいソフトに向いた人はめったにいないよ」とほめられ、それが、ずっとプログラミングを続ける契機となりました。正月に帰省したとき、よしソフトウェアを仕事にしようと決心したことを覚えています。結婚を数ヵ月後に控えた時期でもありました。

3．SPL コンパイラ開発小史

結婚式は 3 月 30 日でした。SPL 開発のための（シ研）派遣は 4 月 1 日からでしたが、4 月 21 日に延期され、それから約半年間戸塚での寮暮らしが続きました。（シ研）は定時退勤が原則ですので定時まで目いっぱい設計に没頭しました。設計の進め方は会議が 3 日に 1 回のペースで開催され毎回、前回に課されたテーマ、例えば「変数名表を設計せよ、if 文のオブジェクトを設計せよなど」で、毎日がかなりハードでした。半年後に工場に移り、定時で帰らなくて済むようになり、かなり精神的に楽になったのを思い出します。設計者としての充実感はこの半年が後にも先にもこの時が一番という印象に残っています。

開発作業は人手が足らず東京のソフトハウスから数人ずつスポット的に派遣されてきました。自分の担当範囲のプログラミングと派遣者へのプログラム仕様の説明と工程管理で目まぐるしい日々が続き夜勤も多かった記憶があります。

開発内容では言語仕様の中に特殊な仕様として問題向き言語を作り出す機能が入っていましが、かなりアルゴリズムが複雑で、コンパイル時間もかなり要しました。そこで当面は使わないということで、開発途中で削除した記憶があります。

最初に動いたときはコンパイルスピードが遅く実用には耐えないものでした。数か月でやっと実用に耐えられるレベルに達しましたが、更なるバージョンアップとしてコンパイラの高速化に数年を要しました。

技術編*

＊本編は筆者の記憶に基づく**開発メモ**です。40 年も前のことで記憶違いや勘違いがあるかもしれませんので、その場合はご容赦願います。正式な論文は野木兼六氏、中所武司氏、林利弘氏、森清三氏のものが情報処理学会論文集や日立評論に収録されているのでそれらを参照して下さい。

1．SPL コンパイラ実装上の課題と解決技術

SPL コンパイラは通常のコンパイラと大きく異なる特徴がある。

1．1　制御用に特有のグローバルデータの課題

鉄鋼や自動車の生産ラインの制御プログラムを作成する場合、システム全体に共通なデータ（グローバルデータ）が数多く存在し、このデータ設計をまずおこなわなければならない。この分野のプログラミング言語はＦＯＲＴＲＡＮにビット操作を追加した工業用ＦＯRTRAN が当時主流であった。しかし、ＦＯＲＴＲＡＮはシステム共通データを直接扱う方法がなく、ＣOMMON データも各サブルーチンの中に個別に書かねばならない。したがって、共通データを先に設計記述して、その後で手続き部分を設計記述するという制御分野の特有の開発方法に、ＦＯRTRAN の記述方式は合っていなかった。当時の解決方法は OS（これも制御用特有）の支援を受けたグローバル型という変数をプログラム中に記述することでかろうじて乗り切っていた。

そこで言語設計者の野木氏、制御の問題に精通した林氏は図 1 のようにデータを記述する宣言文の部分（宣言モジュール）と手続き部分（処理モジュール）を分割して記述し、各々をコンパイル、登録できるように仕様を定めた。さらに宣言モジュールはシステム全体を記述するレベル、サブシステムを記述するレベルなど階層構造をとれるようにした。これによりグローバルな宣言文が一元的に管理できるようになった。

●SPL のプログラム構造

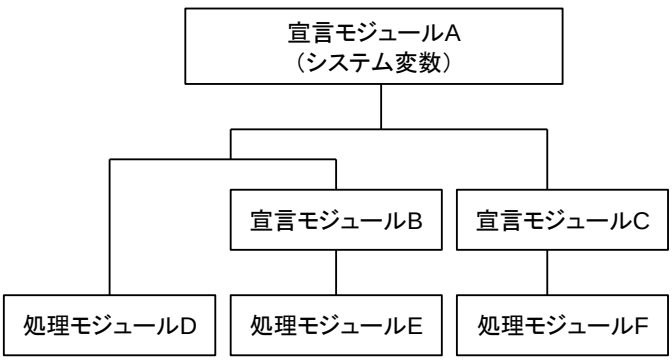


図1　モジュールの階層構造

１．２　浮上したコンパイラ上の技術課題

システム全体の共通データは数１０００個にも及び、処理モジュールのコンパイル時には上位にある宣言モジュール内の変数が必要になるために、技術課題として階層構造の宣言文をいかに効率よく取り込むかが重要なポイントとなった。

１．３　課題の解決策

宣言文は通常のコンパイラではコンパイル中に名前表（テーブル）へ登録される。高速化のために、この名前表をそのまま補助記憶装置に記憶し、次に処理モジュールのコンパイル時に、すぐに呼び出せる方式を設計した。例えば、図２に示すように①宣言モジュールＡをコンパイルした時に、名前表を専用データベース（SPL ライブラリと名付けた）に記憶する。次に②処理モジュールＤのコンパイル時に記憶してあった宣言モジュールＡを読み込む。高速化のポイントは、名前表と一緒に検索用のハッシュ表も記憶することで、しかも階層構造の場合は階層上位をリハッシュして取り込んだ形で記憶するところにある。

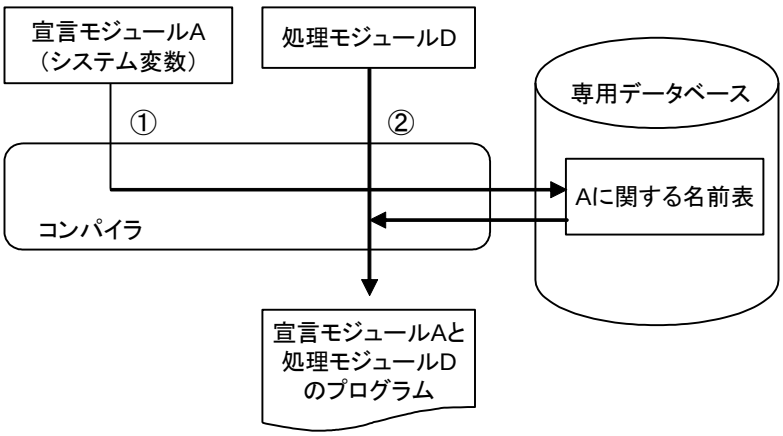


図2 コンパイラの処理方式

通常のコンパイラはこの当時は FORTRAN が主流でコンパイル時に中間情報を補助記憶装置に格納することはない。COOMON データのような共通データは外部表という形でコンパイル結果に不表として付けて、リンケージエディタでデータを結合した。

実装上コンパイル時に補助記憶装置とのデータのやり取りが行われる方式は、環境モジュールを修正した時に発生するバージョン管理がコンパイラの複雑性を増し、不良発生の原因となった。このため数年間はバグの修正に追われることとなった。

１．４　最近の技術水準からみると

当時のソフトウェア開発を取り巻く環境はハードウェア（コンピュータ及び周辺機器）の速度が大変遅く、特にソースカードを読み込み解析するのに時間がかかった。一度コンパイルした中間情報を補助記憶装置に記憶し、再度ソースで読み込むことを避けたのは SPL ライブラリ方式のもう一つの理由だった。ソースカードを補助記憶に格納して再読み込みする方式に比べコンパイル時間の大幅な節約となった。

また、OS は制御用計算機専用の OS（PMS といった）で UNIX はまだ日本では普及していなかった。したがって、OS の支援するファイル構造はなく自前開発となった。

プログラミング言語も、Pascal が教育用言語としてそのエレガンス、有効性が認知されていたが、実用性は疑問視されていた。C 言語はまだまだ日本では使われていなかった。信頼性の高い国防省の言語 Ada はまだ検討段階であった。

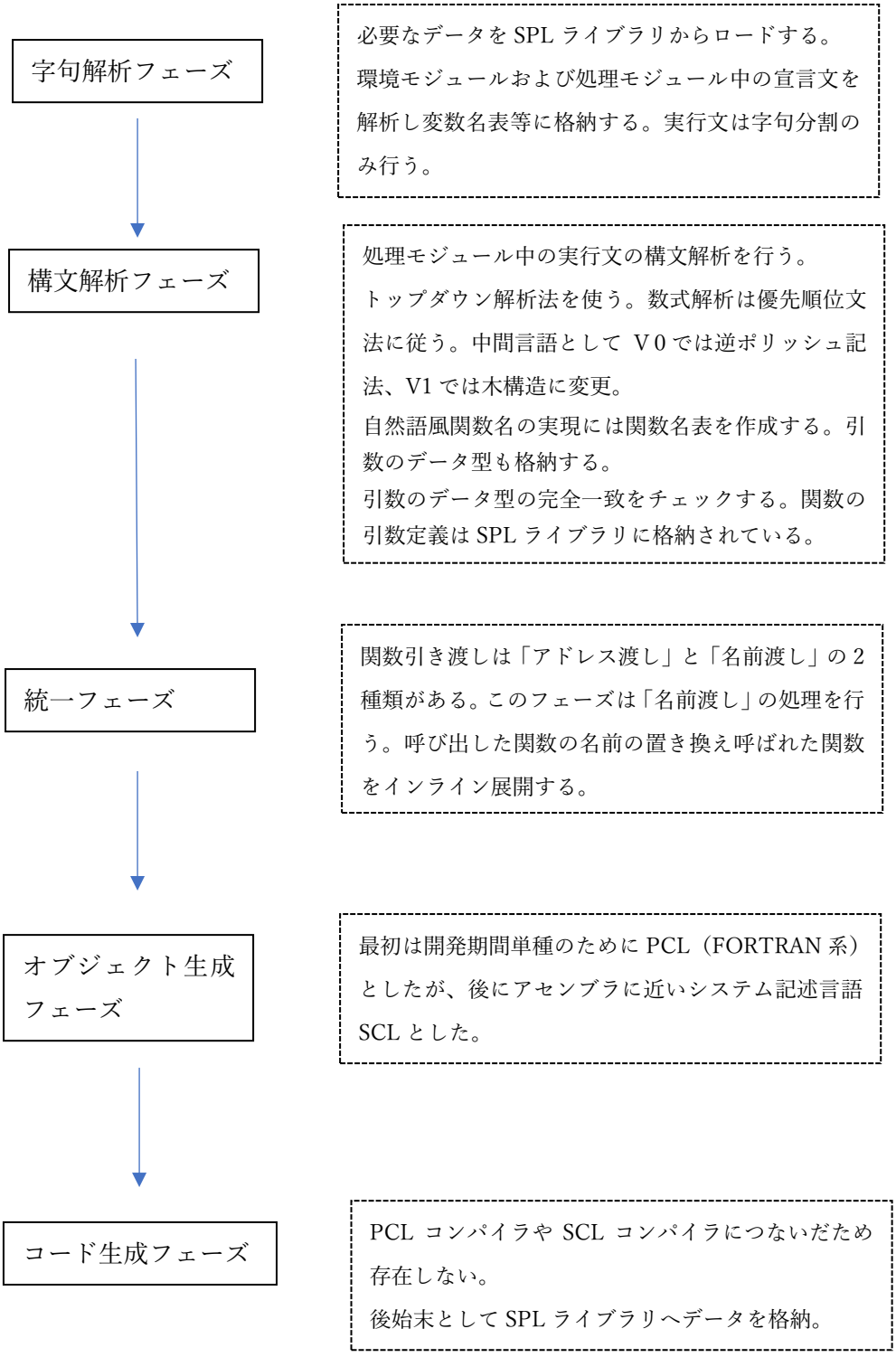
UNIX と C 言語が普及し上記のコンパイル方式は、C 言語の include 文を用いたソースレベルでの管理方式で現在では実現できる。計算機の処理速度の向上した現在では、名前表（バイナリ形式）を記憶する方式もソースレベルを記憶する方式も速度的には差がない。したがって、現在ではソースで目に見える形での管理ができる C 言語の include 文の方が管理はしやすい。しかし、SPL では関数の定義は関数そのものから切り出しているため、常に最新の関数定義となり完全に一元化されている。C 言語では include するファイル中の関数定義は実際の関数定義と一致しているか否かはユーザの管理にゆだねられる。

１．５　SPL の言語仕様の補足

SPL はブロック型言語ではない。文は FORTRAN の宣言文、実行文という言い方が踏襲されている。この辺りは時代の制約と言える。ポインタ変数は PL/I タイプ（特定の変数と結びついる）で C 言語の持つ不定値が発生するポインタ変数ではない。

実引数と仮引数のデータ型は厳密なチェックが行われ、信頼性向上に多大な役割を果たした。

2．　SPL コンパイラのフェーズ構成



※最適化フェーズは作成せず各フェーズ内に組み込んだ。グローバルな最適は行わなかった。出来る限りソースイメージを残した。