プログラム パブリッシング システム

4L−5

木原一夫、綿木輝、相沢栄男、芝野耕司
東京国際大学商学部経営情報学科

1. はじめに

システム開発では、実際のプログラム開発以上に、多くの関連文書の作成が要求される。こうした文書作成の必要性は、プログラムの作成よりも保守が多いこと及びプログラムリスト自体の読み難さに起因する。

また、開発の各段階においても、テスト・検討を完成まで何度となく繰り返し、更にシステムが完成・起動してからも、問題点の修正・改善が繰り返し行われる。この間常に、各段階毎に文書を事細かに作成しているのが現状である。

その結果、（1）文書の大量発生を招き、（2）文書作成に長時間を費すという問題に加えて、（3）プログラムと文書の内容が食い違うという問題が発生している。

これらの問題は、すべて一つの情報がプログラムリストと文書の二つに記録されることから発生する。そして、問題解決の方法は、この二つに分かれた情報を一つに統合することにある。

このような現状を解決すべく研究中のプログラム・パブリッシング・システム(ＰＰＳ)について報告する。

以下、2．では、これまでの関連研究を振り返り、本研究の位置付けを試みる。3．では、ＰＰＳシステムの出力設計について述べ、4．では、システム構成について述べる。最後に5．で、ＰＰＳシステムの今後の課題について述べ結びとする。

2．プログラムリストと文章の統合

プログラムリストと文書を統合したシステムの主な研究には、ドキュメントを中心に統合するドナルド．クヌースの Web システム[1]とプログラムを中心に統合するアーロン．マーカスのアプローチ[2]がある。

クヌースのWebは、書籍の中にプログラムリストを取り込み、統合することをねらったものであるが、一般の開発の現場で、“プログラマ”であっても、“著者”ではないことが多い開発者に適用することには向いていないと思われる。

また、最初に述べた“プログラムリストが読み難い”という問題も再考の必要がある。プログラム言語の設計では、問題記述言語として言語を設計している。事実、ALgol、Pascal　等は、アルゴリズムの記述用として開発され、APL は、IBM System 360 のハードウェア記述用として開発され、実際のコンパイラやインタプリタの開発は、かなり時間が立ってから行われたことは周知の事実である。一方、自然言語での記述には、曖昧性が入る。

すなわち、本来、プログラムリストは、システムの記述文書として望ましいものであるはずである。

それでは、“なぜプログラムリストは、読み難い”のであろうか？この理由として、（1）まず、プログラムリストが処理の正確な記述ではあっても、作成意図や背景の記述にはなっていないことが指摘できる。（2）さらに、一般の書籍に比べると情報をより分かり易く伝える技術としての“組版”がプログラムリストの打ち出しには欠けている点も指摘できる。（なお、手書きの関連文書では、“組版”は行われていなかったが、システム手帳のような様式化によって、この問題点を克服していた。）

我々が今回開発したシステムは、マーカスのアプローチを基に、改良と拡張を行う。

3．ＰＰＳ—システムの出力設計

3．1　マーカスアプローチの評価

マーカスのアプローチ[2]は、次のように要約することができる。

(1)プログラムリストを主に構成。すなわち、付加的な必要情報は、コメントの形で、ソース中に埋め込まれることを想定し、コメントの扱いに関しては特別の配慮をする。

(2)C言語の構成要素(Language Constructs) をよりよく表現するための組版規則の設定。

(3) ページ体裁の設計

このような考慮をもとに、マーカスは、pcc (Potable C compiler) の構文解析をもとにしたプログラムリストの出力整形系を作成した。

このアプローチのメリットは、（1）プログラムと関連文書を統合的に扱うことができ、（2）

Program Publishing System
Kazuo Kihara, Akira Wataki, Eio Aizawa, Kohji Shibano
Tokyo International University

プログラム言語の形式記述能力を組版技術を組み込むことによってより一層高め、結果として、（3）読み易く、正確なシステム文書の作成を可能にした点にある。

しかし、このマーカスのアプローチにも、幾つかの拡張及び改良が必要とされる点がある。以下、プログラムパブリッシングシステムで行った拡張点を述べ、ついで、改良点を述べる。

3．2　ＰＰＳ—出力設計の拡張機能

PPS システムの出力例を図に示す。

3．2．1　システム開発環境全体への配慮

マーカスのアプローチは、プログラムリストのみを対象としている。しかし、実際の開発環境中では、コンパイラが処理するソースコード以外に、さまざまなユーティリティプログラムを用い、それらのプログラムに対する入力情報も、ソースコードと同様に情報を保持している。

この点を考慮に入れ、プログラムパブリッシングシステムでは、make情報の取り込みを試みた。

3．2．2　コメント種別の拡張

マーカスは、行間コメントと行内コメントの二つにコメントを分け、行間コメントに関しては、その出現場所の情報のみをもとに処理法を決めている。また、行内コメントに関しては、一律の扱いをしている。

この扱いは、基本的には、位置情報のみをもとに、決めることになるが、これだけでは、すべてのメタ情報を扱うことはできない。

これに対し、意味ラベル処理を追加した。コメントをマーカスは、二つに分類し、この分類とコメントが位置情報のみをもとに、その出力形態を決定している。

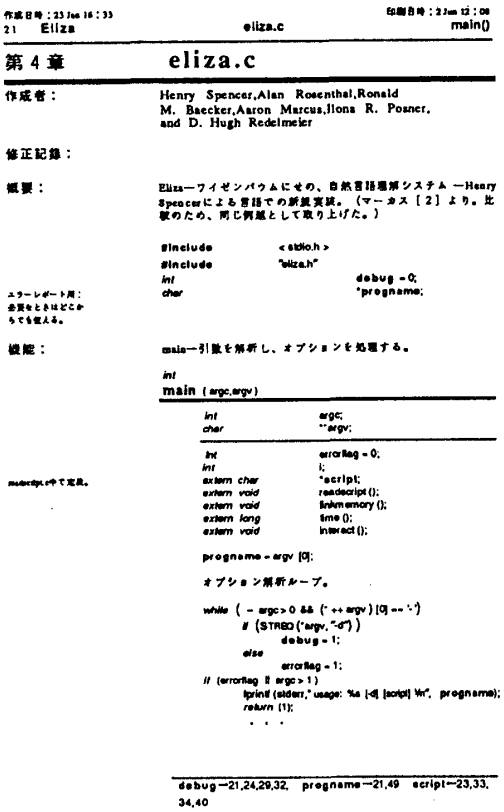
3．2．3　その他の拡張

上記以外に、追加した機能は、より親しみのあるスタイルとして、コマンドリファレンス風のスタイルを採用した。

また、マーカスアプローチにない一覧表示の追加例として、include file一覧、変数とその有効範囲、代入、参照箇所等の情報を網羅した変数一覧等を提示する。

4．ＰＰＳ—システムの構成

ＰＰＳで処理されるプログラムリストは、まず、(1)makeファイルの解析を行う。次に(2)yacc、lexで



定義されたＣコンパイラ部に取り込まれて、構文解析を行い、さらに、ポストスクリプトプログラムを生成するための処理をあらたに組み込む。さらに、構文解析をし、ポストスクリプト命令を組み込んだものを、(3)ポストスクリプト出力生成部に渡し、付加情報の追加を行い、最終的にポストスクリプトプリンタによってポストスクリプト処理され、提示される。

以上が、ＰＰＳ—システム構成の概要である。

5．ＰＰＳ—今後の課題

まず、スタイルのより一層の洗練が今後の課題である。

また、今回はmakeファイルのみの解析を追加したが、SCCS(Sourse Code Control System)、RCS (Revision Control System)等のファイルの情報も含め、プログラムに関するすべての情報を統合することを検討している。

さらに、意味タグの扱いに関しても、拡張を考えている。

参考文献

[1] 有澤誠編、クヌース先生のプログラム論、共立出版、1991.

[2] R.M. Baecker, A. Marcus, Human Factors and Typography for More Readable Programs, ACM Press, 1990.