5E−4　オンライン業務プログラム 移植性検証ツールの適用評価

堀内芳男[1)]，桶村心平[1)]，小宮綾子[2)]
1)NTT 情報通信処理研究所　2)NTTソフトウエア(株)

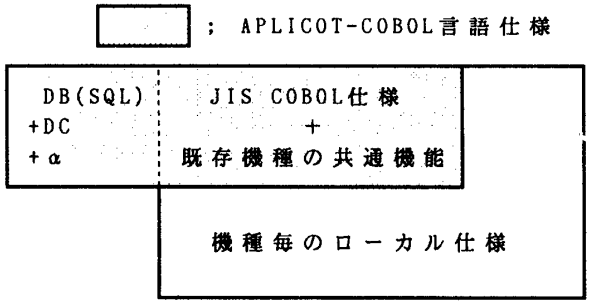
1．はじめに

近年、ソフトウエアの開発量が急激に増加しており、供給量を確保するために既存ソフトの流通性保証が重要な課題となっている。しかし、オンライン業務プログラムのAPインタフェースは標準化が遅れており、機種対応に開発を行わざるを得ないのが現状である。このため当研究所では、APインタフェースを統一して、流通性を高めることを目的としたデータベース（DB）／データ通信（DC）パッケージ（APLICOT[*1,(1)]）の開発を進めている。APLICOTでは、APの流通性を高めるために、APインタフェースの移植性を疎外する要因を制約した仕様を規定（APLICOT−COBL言語仕様）するとともに、仕様に合ったコーディングが行われていることをチェックするための移植性検証ツール（COACH[*2]）を提供している。本稿は、APLICOTのAPにCOACHを適用して得られたデータを基に、流通性を高めるために仕様を制約して、言語仕様を規定する際の留意点、およびCOACHの適用性についての考察結果を報告する。

2．APLICOT−COBOL言語仕様

COBOL言語仕様は、JIS、ANSI、ISO等において標準化されているが、同一機能の書式のサポート範囲の差や、標準化年度の差等により、計算機種によって、実現している仕様が異なっている。このため、ある機種で開発したAPを他機種に移植する場合には、言語仕様の差異を明確化してソースプログラムを書き直す作業が伴うことから、機種間におけるプログラムの流通の疎外要因となっている。APLICOTではこのような問題点を解決するために、APLICOT上のAPに対して、流通性の高いCOBOL言語仕様を提供している。APLICOT−COBOL言語仕様は、JISのCOBOL仕様を基本にして、いくつかの既存の機種で共通に使用できる機能を抽出し、かつDB／DCインタフェースをも規定した、新たなCOBOL言語仕様である。DBインタフェースとしては、標準SQL言語をサポートしており、DCインタフェースとしては、APLICOTがサブルーチンを提供している。

APLICOT−COBOL言語仕様の概念を図1に示す。



α；日本語機能、ブール機能

図1．APLICOT-COBOL言語仕様の概念

3．COACHの概要

COACHは、APがAPLICOT−COBOL言語仕様（図1の網掛け部分）通りに記述されているかをチェックするための移植性検証ツールである。従って、図1に示す"機種毎のローカル仕様部分"で記述されたコーディングに対してエラーを出力する。

4．COACHの適用結果

（1）COACH適用APの概要

APLICOT上のAPは全て、APLICOT−COBOL言語仕様を基にコーディ

The Evaluation of Portability Check Tool for Online Aplication Program.
Yoshio HORIUCHI[1],Shinpei OKEMURA[1],Ayako KOMIYA[2]
1)NTT Communications and Information Processing Laboratories, 2)NTT Software CORP.

ングすることが原則であるが、ＣＯＡＣＨを適用していない状態のＡＰをそのまま入手した。本ＡＰにＣＯＡＣＨを適用した部分は、表１に示す３モジュール　６５Ｋステップ（１５０ルーチン）である。

表１．ＣＯＡＣＨ適用ＡＰの概要

	ﾓｼﾞｭｰﾙの特徴	規模	ﾙｰﾁﾝ数
ﾓｼﾞｭｰﾙ1	COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ経験者ﾁｰﾑ1が作成	K 30	97
ﾓｼﾞｭｰﾙ2	COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ経験者ﾁｰﾑ2が作成	K 22	32
ﾓｼﾞｭｰﾙ3	COBOLﾌﾟﾛｸﾞﾗﾐﾝｸﾞ初心者ﾁｰﾑが作成	K 13	21

（２）ＣＯＡＣＨ適用結果

表１の全モジュールにおいて発生した、移植性疎外エラーの種別は１７種類であった。本エラー種別は次の２種類に分類される。

（Ａ）移植時にロジック修正が必要なエラー

（Ｂ）移植時にロジック修正が不要なエラー

上記１７種類のエラー種別で、ロジック修正が必要なエラーは、４種類（"一意名に部分参照を使用している"、"ACCEPT 一意名 ･･･の一意名に内部10進形式を使用している"等）であった。また、ロジック修正が不要なエラーは、１３種類（"IF命令にTHENがない"、"表意定数にSPACESを使用している"等）であった。ロジック修正要／不要エラーの発生頻度（ルーチン数）を表２、３に示す。

表２．ロジック修正必要エラーの発生頻度

ｴﾗｰ種別	Ｅ１	Ｅ２	Ｅ３	Ｅ４	計
ﾓｼﾞｭｰﾙ1	13	6	4	0	23
ﾓｼﾞｭｰﾙ2	25	8	2	1	36
ﾓｼﾞｭｰﾙ3	10	0	0	0	10
計	48	14	6	1	69

表３．ロジック修正不要エラーの発生頻度

ｴﾗｰ種別	a	b	c	d	e	f	g	h	i	j	k	l	m	計
ﾓｼﾞｭｰﾙ1	20	18	18	7	0	1	2	2	2	2	0	1	0	73
ﾓｼﾞｭｰﾙ2	5	5	1	2	5	3	0	0	0	0	0	0	1	22
ﾓｼﾞｭｰﾙ3	0	1	0	0	0	0	0	0	0	0	2	0	0	3
計	25	24	19	9	5	4	2	2	2	2	2	1	1	98

5．考察

前章の結果より整理できる、移植性疎外要因エラーの特徴を以下に列挙する。

①ロジック修正が必要なエラーはＣＯＡＣＨ適用ルーチン数の４６％に発生している。

②モジュール３はモジュール１、２に比較して、ロジック修正要／不要エラーともに発生頻度が小さい。

③ロジック修正不要エラーは、エラー種別ａ～ｆで発生頻度の約９０％を占めている。

④同一レベルのプログラミング経験者が作成したモジュール（１、２）でも、チームが異なるとエラーの発生頻度に差が生じる。

上記特徴から、以下のことが考察される。

（１）ロジック修正要のエラーが、全ルーチンの５割に内在し、ドキュメントのみでは除去されていないということは、移植性検証ツールの有効性、および必要性を立証している。

（２）②の理由は、初心者の方が言語仕様書（ドキュメント）をよく読んでからコーディングするので、新しい仕様（移植性疎外要因の制約）によるコーディングミスを起こしにくいためと類推できる。

（３）③、④より、コーディング開始時、全てのチームに対して一様に、発生し易いエラーの周知徹底を図れば、エラー発生頻度を均一化して大幅に削減できる。

6．おわりに

移植性検証ツール（ＣＯＡＣＨ）により得られたデータから、以下のことが明かとなった。（１）移植時にロジック修正が必要となるエラーは、全ルーチンの約５割に発生している。（２）初心者よりむしろ、経験者の方が、移植性疎外要因エラーを起こし易い。（３）ＣＯＡＣＨを提供しない場合も、コーディング開始時、発生し易いエラーの周知が重要である。

*1 Application PLatform for Integrated database & data COmmunicaTion

*2 db/dc CObol Application program CHecker

＜参考文献＞(1)NTT情報研発行「APLICOTﾏﾆｭｱﾙ・ｼﾘｰｽﾞ」　(2)JISﾊﾝﾄﾞﾌﾞｯｸ情報処理ｿﾌﾄｳｴｱ編'88