業務アプリケーション開発時における部品化の検討と適用結果の評価

3D－5　古川　純子　鵜澤　亨　降旗　由香理
（株）日立製作所　公共情報事業部

1．はじめに

システム開発で重要視される「生産性の向上」と「保守の容易さ」を実現する一つの方法が部品化である。これを徹底するには、業務ロジックの部品化を図ることが必要である。実践においては、設計時の部品化すべき業務ロジックの切り出し、アプリケーション開発時の作成した部品の管理、開発への展開の仕方などが成功するための鍵となる。

本論文では、米 PowerSoft 社の PowerBuilder を用いた財務会計システムの開発における業務ロジック部品化の試みを紹介をすると共に、適用結果の評価を報告する。

2．部品抽出

部品抽出は次の観点から行った。

① 複数の画面に共通のレイアウト、ロジックを持つ入力項目（表１－１～５）
② 複数のプログラムで使用されるロジック（表１－６）
③ 項目選択用ウィンドウ（表１－７）

①は幾つかの入力単項目を集めたもので、入力チェックや入力値に対する名称取得等業務ロジックを含めユーザオブジェクトとして作成した。②は関数として、③はウィンドウとして作成した。作成した部品を表１に示す。この中で項番 1～5 は特に伝票入力系プログラムでの部品化をターゲットとして作成したものである。伝票入力系プログラムは全 290 本中 69 本あり、比較的似た画面レイアウトと業務ロジックを持つ。

例として項番１の歳出科目入力部品について説明する。これは、入力フィールドにコードを入力するとDBから該当する名称を取得し表示する部品である。

表 1　部品一覧

項番	分類	数	部品名	機能
1		1	歳出科目入力	
2	伝票入力用	1	歳入科目入力	コード入力すると対応する名称
3	集合項目部品	1	相手方入力	をDBから取得し表示
4	(ユーザオブジェクト)	1	摘要入力	
5		2	日付入力	日付を和暦で表示
6	単項目部品	97	関数	DB検索関数、文字列編集関数他
7	ダイアログ部品	50	ズーム	DBから名称等を取得し一覧表示
	合計	153		

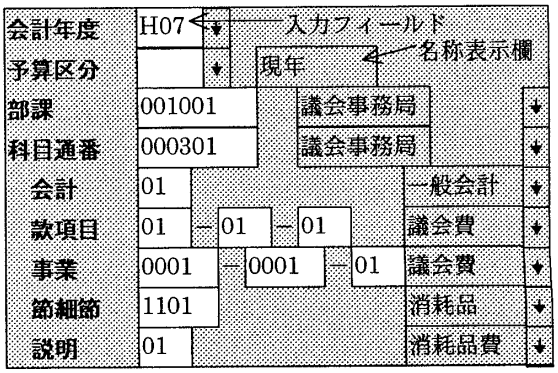

図 1　歳出科目入力部品

本システム開発ではこのような集合項目部品をまず特定の業務プログラム上に埋め込んだ形で同時開発を行い、後続のプログラムに再利用する手順で全体の開発を行った。

3．適用結果

本論文では業務ロジックの部品化に論点を絞り、表１の項番１～５の部品を再利用したプログラムの作成工数に着目して評価分析を行う。

3．1 作成工数の分析

図２は部品開発と並行して作成を行ったプログラムＡと、ここで作成した部品を再利用して作成した４本のプログラム（Ｂ～Ｅ）の作成工数比のグラフである。プログラムＡの作成工数を 100 として相対数値で示した。

Evaluation of an examination of parts and an application result at the time of business application development
Junko Furukawa,　Toru Uzawa,　Yukari Furuhata
Hitachi,Ltd.　Government & Public Corporation Information Systems Division

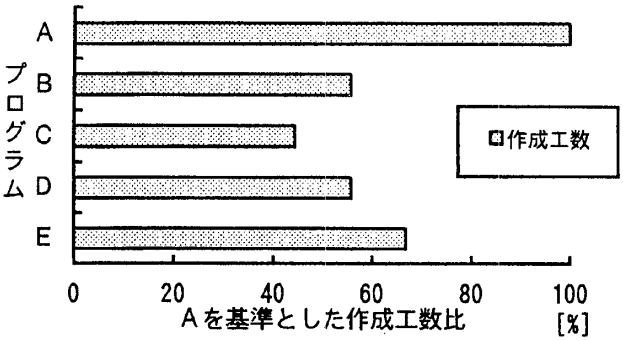

図 2　部品適用プログラム作成工数

各プログラムにはそれぞれ3種類の業務部品を使用しており、画面上の入力項目の大半が部品上に含まれている。部品の再利用により工数は、約50%削減できた。

但し、工数削減率の50%という値は画面上の入力項目の大半を部品化した割には小さいと思われる。そこで、各担当者による追加コーディングの内容を分析した。

3．2　コーディング内容の分析

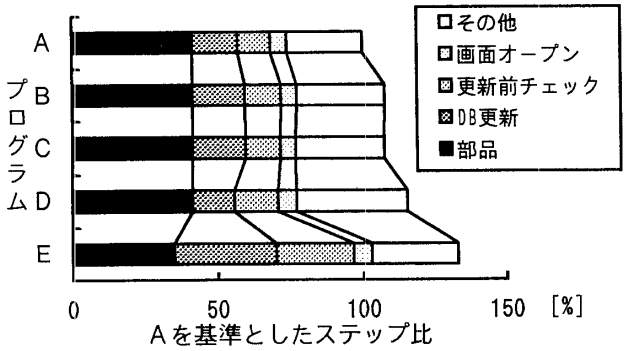

図 3　ステップ数の割合

ソースコードについて部品化部分とコーディング部分に、コーディング部分はその内容により更に分類し、構成ステップ数の割合を調べた。その結果を図3に示す。図2と同様にプログラムAの総ステップ数を100として相対数値で示した。図3から、部品を再利用しても各プログラムともに全体の50%強のスクリプトは新規に記述されていることが判る。ここには画面レイアウトの様な明確な類似性を見つけにくいものの、DB更新や更新前チェックなど共通化可能な業務ロジックが潜在していることが判った。

また、各担当者が部品提供者との同意なしに自分の必要に応じた部品作成をすすめている場合も多々あることが判った。例として、金額入力のためのユーザオブジェクトは37個作成されている。しかし、これには機能の重複したものが多く、実際には小数の金額クラスに集約できることが判った。

4．結果の分析

コーディング内容の分析から明らかになった問題点に対する対策を検討した。今回の業務部品抽出は画面レイアウトの類似性といった特定の面から進めたために、共通化の範囲に偏りができてしまった。また、全体開発時に各担当者が作成する業務部品を管理していなかったために重複開発が発生することになった。前述した問題は、各担当者の部品追加のベースとなる広範囲な基本クラスの提供と、作成した部品を共通ライブラリにフィードバックし再利用を促進する体制を整えることで防ぐことができると考える。

ここで、共通化が可能と考えられる部分を部品化した場合に、図3の各プログラムのコーディング内容がどのように変わったかを調査した。その結果を図4に示す。

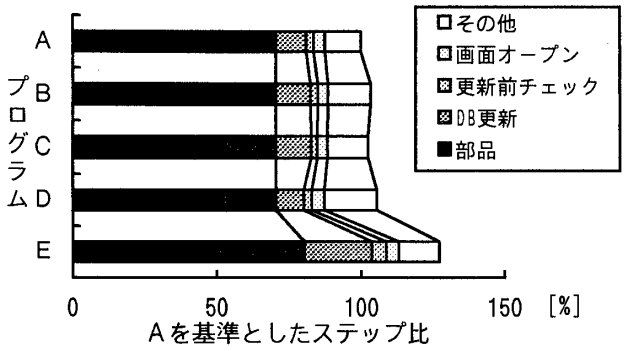

図 4　対策後のステップ数の割合

図4よりスクリプトに占める部品の割合は約50%から約70%に上がるという結果を得た。

5．おわりに

部品化による効果を確認できた反面、部品の管理、適用の方法における問題点や課題も明らかになった。さらに、部品の再利用率を高めるためには、部品設計の当初から広範囲な部品化を検討することの必要性が明らかになった。本論文での結果を踏まえて、今後さらに生産性の向上を推進していきたいと考える。