ウィンドウシステムのプログラム移植作業における ソフトウェア工学的評価

4S－1

平井孝史，小島大吾，玉山尚太朗，早川栄一，並木美太郎，高橋延匡

（東京農工大学）

1. はじめに

われわれの研究室には、ペンの UI の研究基盤として独自に開発されたウィンドウシステムである“未”が存在する。“未”は 1993 年に開発されたあと、われわれの研究室の在籍者によって保守・改良され、現在も稼動している。

ここで、現在の“未”においては、動作するハードウェアやその発色数等の問題があり、これらのことから、“未”の移植を行った。

本稿では、この“未”のプログラムを移植した際に発生したバグをまとめたものについての考察と“未”の評価を述べる。

2. 移植作業

(1) 分担

移植作業はA・B・Cの三人で、Aはハードウェアに実際に描画をさせる描画ドライバとそれを扱うためのライブラリ。Bは“未”のカーネルとマウスドライバ。Cは“未”の UI マネージャと AP が“未”を使用するためのライブラリとフォント部分。このように担当させた。

(2) 作業期間

作業期間は次のとおりである。

1994-09	移植プロジェクト開始
10	移植の方針・変更点・分担　決定
11	ここから各自の作業に入る
	元ソースの理解
	新ソースの作成
1995-05	各自の作業終わり
	デバッグ開始
06	完成

移植のプロジェクトが動き始めたのは 1994 年の 9 月であり、それから各自の担当の作業に入ったのは 2 ヶ月後であった。

各自のソースの移植が終わり、リンクしデバッグを開始したのは 5 月の終わりである。それからデバッグの終了までにおよそ 10 日かかった。

3. バグの統計と考察

ここでは、移植の際に発生したバグについての統計データとそれについての考察を述べる。

3. 1 ソースの再利用率

ソースの再利用率

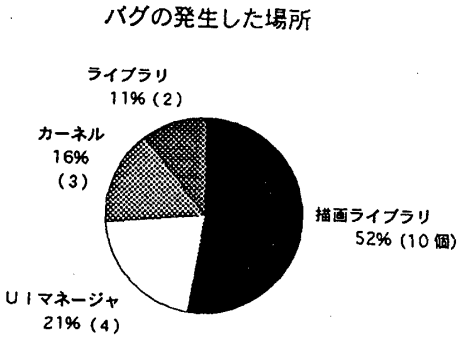
	ヘッダファイル	ソースファイル
A	175/868 (20 %)	760/3300 (23 %)
B	704/1863 (38 %)	11368/13476 (84 %)
C	1520/2456 (62 %)	9394/10867 (86 %)

ソースの再利用率はAの担当分の 23% が最も低い。これは、Aの担当分である描画ドライバ・ライブラリはハードウェア依存度が高く、ドライバは完全に新規作成であるし、ライブラリに関しても以前の関数名で使えるように“未”の形式にあわせるところ以外は新規作成を行ったためである。つまり再利用は以前のものとのインタフェースをとるためだけに行われた。ハードウェア依存部は再利用がしにくいといえる。

“未”の本体を移植したBとCはどちらも 80% 以上の再利用率となっている。これは OS 依存部分以外はほとんど再利用できたためである。これらはどちらも 10000 行を越えているが、行数の規模に比べ移植の作業は容易に行えた。

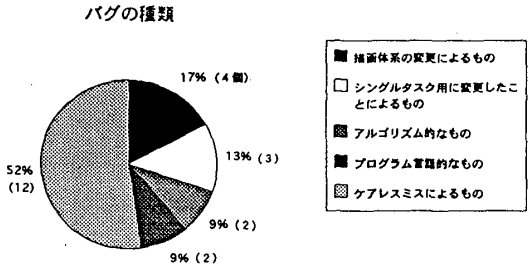
3. 2 バグの発生した場所

Software Engineering Evaluation of the Transplantation of a Window System Program
HIRAI Takashi , KOJIMA Daigo , TAMAYAMA Syoutarou , HAYAKAWA Eiichi , NAMIKI Mitarou and TAKAHASHI Nobumasa
Tokyo University of Agriculture and Technology



バグは描画ライブラリに 10 件発生し、52%（10 個）と全体の半分を占めている。先のソースの再利用率からもわかるように、これは描画ライブラリが、カーネルや UI マネージャのような単純な移植作業よりも、新規に作成した部分が多いことが原因として考えられる。

3. 3 バグの種類



一番多いのはケアレスミスで、52%（12 個）と全体の半分を占めている。そこで、これらのケアレスミスをもっと細かく分析してみる。なぜこのようなケアレスミスが起きたのかを考えてみた。

これらがなぜ起きたのかを表にまとめてみると次のようになる。

バグ	原因
・未定義シンボル	三人の連絡の不徹底
・マクロの使用ミス ・インクルードファイルの変更ミス ・ID の二重変換 ・free() ミス ・関数の返値のミス	設計をせずに変更・作成
・オブジェクトファイルの転送ミス	開発環境
・アイコンデータの変数型宣言忘れ ・変数の代入ミス	ケアレスミス
・渡す引数のミス	変更前のソースに存在

このようにしてみると、ケアレスミスが原因で起きたバグも、「三人の連絡の不徹底」や「設計をせずに変更・作成」などのバグは未然に防げたものだと考えられる。そうすればケアレスミスの数は半分に減らすことができたはずである。

4. “未” ウィンドウシステムの評価

ここでは、移植作業を通しての “未” の評価について述べる。

(1) 移植作業のしやすさ

“未” は、カーネルと UI マネージャの二つから構成されており、また実際に描画を行う部分はハードウェア依存部として独立している。内部がこのように独立しているため、作業の分担を行うことが容易であった。分担を行うことができればその分個人の負担が減るため短期間で作業を行うことができる。つまり分担がしやすいことから移植作業がしやすいといえる。

(2) 再利用のしやすさ

“未” は、ファイル単位で見れば、カーネルが 17 ファイル、UI マネージャが 11 ファイルと、その内部が機能単位ごとに細かくモジュール分けされている。

先に述べた再利用率の高さとモジュール分けの細かさ、またシステムそのものがカーネルと UI マネージャの二つに分けられていることから、“未” は再利用が行いやすいといえる。

(3) 保守のしやすさ

“未” はシステム内部が細かく分けられているために、保守はしにくいともいえる。例えば、ウィンドウの管理などはカーネルと UI マネージャの両方で行っているため、うまく動作しない場合にどちらに問題があるのかはわかりにくい。つまり、処理内容に重なりがあるような部分があるため、「保守」という点では問題がある。

つまり、保守しやすいプログラムを書くためには、ハードウェア依存／非依存や機能単位ごとにモジュール分けするのはもちろん、管理機構などは処理内容にできるだけ重なりのないようにする必要がある。

5. おわりに

本稿では、ウィンドウシステムの移植時に発生したバグについての考察と “未” の評価について述べた。

今後は、規模の大きなプログラムを動作させる予定である。

[1] 津田道夫 他：ソフトウェア変更作業の分析と支援機能，情報処理学会，ソフトウェア工学研究会，102-1，1995