マルチプラットフォーム製品のテストについて

3N−1

村上徳宏

(株) 富士通神戸エンジニアリング

1．はじめに

ＵＮＩＸシステムのビジネス分野での普及が進む昨今，ユーザのニーズも汎用機からサーバ，ＰＣ／ＷＳへの分散運用へと移ってきている．当社の開発製品についてもこれらのニーズに応えるため汎用機からサーバ，ＰＣ／ＷＳ上で同一の機能を提供する製品（マルチプラットフォーム製品）の開発が主流となってきている．

私はこれら製品の検査業務を担当している．ここでは，マルチプラットフォーム製品のテストの一手法について紹介する．

2．分析

― マルチプラットフォーム製品とは

マルチプラットフォーム製品とは以下の特徴をもつ製品である．

- 同一機能を各ＯＳ上で提供する
- 設計当初から共通部と固有部を意識している
- 一般的にＣ言語で開発される

開発工程は以下のようである．

① 基本設計で製品の役割・世界情勢・ＯＳの違いを意識した枠組みを決定する．
② 機能設計で具体的な機能を決定する．
③ 構成設計で共通部と固有部の細分化・インタフェースの決定を行う．
④ 詳細設計，コーディングで構成設計にもとずいたソースを作成する．
⑤ 単体テスト，総合テストで共通部，固有部に着目した動作確認を実施する．

― 検査員のテストは

検査員は，外部仕様書をもとに製品をテストし，ユーザ個々の要件を満たせるかどうか，以下の観点で評価する．

- 共通部では機能，運用などに着目
- 固有部ではＯＳ固有部に着目

共通部に関するテストは，外部仕様書をもとにテストケースを抽出し，かつ，テストセットのマルチプラットフォーム化をはかっている．

ＯＳ固有部に関するテストは，対象製品の外部仕様書だけでなく，ＯＳ，ハードに関する外部仕様書，対象製品の内部仕様書（コンポーネント構成など）をもとにテストケースを抽出している．

― 検査員のテスト手法について考えると

マルチプラットフォーム製品のテストは，通常，各ＯＳごとに実施していく．一つのＯＳ上での提供が終わっていれば，共通部に対する基礎品質は高いことになる．おのずとテストの着目点はＯＳ固有部に集中する．

ＯＳ固有部について，検査員があらゆる条件下での動作保証をとるために，対象製品の外部仕様書からだけでは抽出できないテストケースもあることから，ＯＳ，ハードの外部仕様書や対象製品の内部仕様書から，対象製品とＯＳとの係わりに関する知識を得る必要がある．このため，検査員は，テストケースの抽出の前に，調査を実施するが，マルチプラットフォーム製品のようにＯＳの種別が多い製品については，調査にかかる負担が大きくなる．

私は，調査にかかる検査員の負担を軽減するため，ＯＳをあまり意識せずにＯＳ固有部のテストケースの抽出ができないものかと考えた．以下のその手法を紹介する．

The Test of Products on Multi-Platform
Naruhiro Murakami
Fujitsu Kobe Engineerling CO.LTD

3．アプローチ

－ ソースに着目する

製品のＯＳ固有部との関わりは，ソース上の以下のものから把握できる．

・ マクロ
・ 外部関数
・ 標準関数
・ コンパイル・リンク方法

開発員はマクロ，関数の外部仕様書をもとに，呼び出し順番やパラメタなどに注意して設計する．このため外部仕様書の解釈違いにより処理誤りが生じる可能性がある．また，マクロや関数がブラックボックスであるため，例えばマクロや関数の呼び出し方法を共通に使用できるような場合，実はＯＳごとに仕様の違いがあったとしても，これを見落とす可能性が高い．

これらの問題がないことを確認するため，検査員は，これらマクロや関数の呼び出し箇所をソース上で全て認識する．加えて，マクロや関数の呼び出し方法を確認する．テストケースの抽出のための調査を上記の範囲で隈なく実施し，テストケースを抽出する．

ＯＳの違いは，マクロや関数で意識できるため調査する範囲も最低限ですむ．

なお，コンパイル・リンクの方法については全オブジェクトを呼び出すことにより保証は可能であり，ここではあえて述べない．

－ テストの網羅性の検証を行うために

検査員は上記，テストケースによりＯＳ固有部が実際に呼び出されているかどうかを確認するため，呼び出し箇所に標識（通過したことが分かる仕組み）をつけていく．

従来，標識をソースに埋め込む手法は単体テスト時のルートの確認で使用するケースが多い．また，総合テスト時にテストの網羅性検証のため使用する場合もあるが，埋め込み位置の設定や評価が難しく検証の決め手材料とはなっていない．ここでは，ＯＳ固有部に着目しているため，設定の評価が容易である．

マクロ，外部関数については，呼び出し箇所の全てに標識を埋め込む．

標準関数については，一般的に使用頻度が高く，呼び出し箇所の全てに標識を埋め込むことは困難である．標準関数の場合，インタフェースが比較的簡単な場合が多いことから，関数の種別・入力条件に着目し標識を埋め込む．

ＯＳ固有部に着目したテストケースをともに，テストを実施する．標識によりテストが網羅されていることを確認することによりテストの網羅性の検証が容易にとれる．

4．効果

標識の埋め込み手法を用いることにより，以下のような効果がある．

－ テストの網羅性向上
－ テストケースの抽出のための調査工数削減

5．今後の課題

標識を埋め込めば，標識の有効／無効の判定処理がはいるため，テストの網羅率向上の代償として，ある程度のオーバヘッドがかかることとなる．オーバヘッドの削減のため，テスト終了後に自動的に標識を抜き去るような手法の確立を検討していく必要がある．

また，これまで述べてきた手法は，マルチプラットフォーム製品の特徴をとらえた上で，ＯＳ固有部に着目した標識の埋め込み手法を用いたものである．例えば，機能追加のようにベース部分と追加部分が別れているような製品などへの適用についても検討していく必要がある．

以上