ソフトウェア工学 126−10
（2000．3．10）

シミュレーションによる組み込みソフトウェアテスト方式とテスト環境の効率的な運用方式の提案

原圭吾、植木克彦、平山雅之

株式会社東芝　研究開発センター

抄録

近年、家電や携帯端末のような組み込みシステムは大規模化と複雑化が進んでいる。これらの変化に対応し、かつ加速する市場サイクルに追随していくためには、開発工程の効率化と短縮が不可欠である。特にこれらの複雑な組込みシステムではテストに多大な工数を必要としており、開発遅延などを引き起こす大きな要因の一つになっている。これに対しソフトウェアシミュレータを使ったテスト環境を利用する事でテスト工程を効率化する方式が提案されているが、これらの方式ではシミュレーションのための環境を構築するためのコストがかかるといった問題が新たに生じている。

本報告ではシミュレータの再利用を円滑に進める方式を提案し、これにより上記の問題の解決を図るアプローチを採用した。ここではシミュレーション環境を再利用するにあたり、再利用する側の要求機能、再利用される側の機能や再利用のパターンをもとに、シミュレーション環境全体の機能分割について、再利用性を考慮した分割手法を提案した。またシミュレーション環境の再利用を進める際の作業プロセスにも着目し、再利用時の機能抜け防止を目的としたシミュレーション環境構築プロセスを提案した。シミュレーション環境構築プロセスとしては特に再利用するシミュレータに対する機能の実現度合いやシミュレーション精度に着目し、機能の過不足や精度の不足を動作時にアサーションなどを応用して検証する方式についても提案する。

A Proposal of Embedded Software Testing Method Using Simulation, and an Efficient Process to Reuse Software Simulator

Keigo Hara,　Katsuhiko Ueki,　Masayuki Hirayama

Research and Development Center, TOSHIBA Corp.

Abstruct

Recently, it became popular to use software simulators for embedded software testing because it brings high efficiency of testing and also high quality of software. On the other hand, since simulation environments consist of these simulators generally require much building cost and time. Therefore, re-use of existing simulators is recommended. However, conventional simulation environments and simulators have not been considered their reuse mechanism enough, it is difficult to reuse these simulation environments. In addition, the difference between the building environment and the reusing environment often causes functional shortage of the building environment.

This paper proposes an architecture of simulation environment which focuses on frequency of reusing, and building process using assertion technique to detect functional shortage. Proposed architecture and process enable to reuse existing environments efficiently and also to achieve high reliability of simulation environments.

1. はじめに

近年の家電および産業機器の多くはマイクロコンピュータ（マイコン）によって制御されている。これらシステムは組み込みシステムと呼ばれ、その制御ソフトウェアは組み込みソフトウェアと呼ばれる。これらの組み込みソフトウェアでは、システムの大規模化と高機能化に伴うソフトウェアサイズの増大とソフトウェア自身の複雑化が急速に進行している。この一方で製品投入サイクルの短縮に伴う開発ライフサイクルの短期化が進んでいる。これらのしわ寄せは組み込みソフトウェアのテスト工程に顕著に表れる結果となっている。また組み込みソフトウェアはその性質上、テストに試験用ハードウェアや専用の試験装置を必要とする等の問題点を抱えており、組み込みシステムのテスト工程に関する大きな足かせとなっている。これらに対する改善策として、シミュレーション環境を利用することでテスト作業の効率化や品質向上を実現する方式が提案され、実際の開発フィールドでも広く利用されている。しかしシミュレーションを利用したテストではシミュレーション環境の導入構築コストが発生し、コスト面では有意な効果が得られないことも少なくない。これに対し、既存のシミュレーション環境を再利用することで構築コストを削減することが試みられている。しかし実際の開発において利用されるシミュレーション環境は必ずしも再利用性を考慮して構築されていないため、再利用に支障をきたす場合が多い。

また再利用するシミュレーション環境と新たに構築するシミュレーション環境の仕様の違い、あるいは仕様の曖昧さがある場合、最終的に構築されるシミュレーション環境の機能に過不足が発生し、シミュレーション対象のテストに支障が生じる可能性もある。以上のような背景から、シミュレーション環境の再利用性および信頼性を向上させるための手法が必要とされてきている。

本報告では、シミュレーション環境の構成モジュールを他環境の構築において効率的に再利用するためのモジュール構成設計方式と、他環境のモジュールを再利用する際の機能抜けを防止するための環境構築手順の提案を行う。

2. シミュレーションを利用したテストの問題点

組み込みソフトウェアはその性質上、テストに試験用ハードウェアや専用の試験装置を必要とする等の問題点を抱えている。それらを解決し、組み込みソフトウェアのテストを効率化する手段として、シミュレーション環境を利用する手法が存在する。シミュレーション環境は、テスト対象ソフトやハードウェアの動作を模擬するシミュレータ群と、シミュレータの動作可視化ツールのような支援ツール群から構成され、ソフトウェアを PC やワークステーションのような開発環境上で実行および動作解析することができる。こういったシミュレーション環境を利用してソフトウェアの論理動作のテストを行なう事により、テスト工程を前倒しする事ができ、工数の後期集中や待ち人員の発生を防ぐことができる。しかしながら、テストを目的としたシミュレーション環境には次のような問題点がある。

[環境の再利用性]

ソフトウェアテストのためのシミュレーション環境を構築するにはシミュレータ作成のためのコストが必要となる。通常、シミュレーション環境の構築時には、汎用シミュレータを購入したり、類似製品のシミュレーション環境を流用することでコスト削減を図る。しかし実際にシミュレーション環境を流用する際には、次のような問題点が発生する。

- 汎用のシミュレーション環境は操作性・作業効率の面で使いづらい
- 利用目的の違いから、必要な機能が備わっていない可能性がある
- 製品独自の機能に関する部分は、基本的に既製品に無い

従って、不足している部分や変更が必要な部分に関しては、利用者が新たに実装を行なわなければならない。しかし、新規にシミュレーション環境を構築する際には必ずしも再利用性まで考慮して作成していない場合が多く、再利用の際にモジュールの切り出しや差し換えが容易に行なえないのが現状である。また、カスタマイズ時にバグがはいることも少なくない。

[機能の抜けの発生]

既存のシミュレーション環境を再利用する場合、再利用元のシミュレーション環境下ではある機能が実現されているように見えても、その機能が再利用した先の環境で必ずしも期待したとおりに動くとは限らない。これはシミュレーションを利用したテストでは、対象システムの動きを完璧に模擬実行することが求められるケースは少なく、テストすべき項目に対応させて必要最低限のシミュレーションの機能や精度しか実現していないことに起因している。このため、再利用するシミュレーション環境の仕様と新たに再利用によって構築しようとしているシミュレーション環境の仕様の間に違いがある場合、再利用して構築したシミュレーション環境で必要な機能が実装されていなかったり、ごく簡略化した動作しかできないといった状況が発生する恐れがある。このようなシミュレーション環境の機能抜けが存在すると、シミュレーション対象の正確かつ十分なテストが出来なくなり、製品の品質の低下に繋がる危険も存在する。

3. 再利用を考慮した環境の構成

本節では、シミュレーション環境の開発コストを抑える事を目的とし、効率的な再利用を可能とするためのシミュレーション環境の構成について述べる。ここでシミュレーション環境が提供する機能を、

- 対象ソフトウェアの動きや周辺の外部環境や周辺システムの動きを模擬実行するシミュレータ機能
- 対象ソフトウェアへの入力や実行結果の出力などを行なうテスト支援機能

の2つにカテゴライズし、これら機能を提供するアプリケーションの集合をシミュレーション環境と定義する。通常のシミュレーション環境では対象システムの動作を模擬するために、複数のシミュレータ機能と複数のテスト機能から構成される。また、個々のシミュレータ機能はそれを実現する一つもしくは複数のモジュールに分割される。これらの関係を図1に示す。

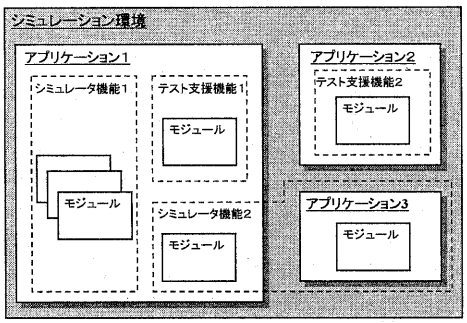

図1:シミュレーション環境構成の定義

3.1 再利用パターンの分析

シミュレーション環境の再利用を効率的に行なえるかどうかは、再利用時の修正がどの程度の規模で発生しうるか、およびどの程度の頻度で発生するかに大きく依存する。従って新規にシミュレーション環境を構築する際に、将来にわたりどのような修正が発生するのかを想定しておくことが重要となる。一般にシミュレーション環境の再利用パターンとしては以下のようなものが考えられる。

(i)シミュレーション環境の利用形態変更

このケースではシミュレーション環境が提供する機能のうち、主としてテスト支援機能に関して、その利用形態が変更される場合を想定する。これは例えば手動でデータ入力と結果確認を行なうためのテストから、自動で大量のテスト実行を処理するためのテスト環境を構築する場合などが相当し、このケースでは全てのシミュレータに対して入出力モジュールの変更が必要となる等、大規模な修正が発生する可能性がある。この変更は実行ごとに発生する可能性があり、変更頻度は高い。

(ii) シミュレーション対象の機能追加／修正

このケースはシミュレーション環境が提供するシミュレータ機能に関する変更が主として考えられる。シミュレータ機能に関する変更は模擬実行する対象の製品ソフトウェアや周辺システムや外部環境の変更などによって引き起こされる。

例えば製品Aの動作を模擬実行するシミュレータ機能に対し、機能追加された製品A'の動作を模擬実行するといった場合がこれに相当する。このケースでの変更部分はシミュレーション対象の製品や外部環境・周辺システムなどの追加/変更機能に対応したモジュールに限られ、変更の発生頻度も製品のバージョンアップ一度に対し1〜数回とそれほど頻繁には発生しない。

(iii) シミュレーション対象の変更（同分野）

既存の製品Aのシミュレータを再利用（流用）し、同分野の製品Bのシミュレータを構築する場合などに相当する。変更規模は、製品AとBの間の類似度に依存する。発生頻度は、製品開発立ち上げ時に1〜数回。変更範囲は主に製品の機能単位の変更が発生する。さらに、各モジュール間のインタフェース変更も発生する可能性が高い。

(iv) シミュレーション対象の変更（異分野）

既存の製品のシミュレータを流用し、動作の大きく異なる異分野製品Bのシミュレータを構築する場合などに相当する。一般にはこのような再利用はほとんど発生しない。また製品間の機能仕様のギャップが大きいことが考えられ、シミュレーション環境の再利用という視点からは外れるケースが少なくないと考えられる。

ここで、変更範囲と変更頻度が大きいほど、再利用時の修正コストが大きくなる。従って、これらのコストの大きな変更パターンを優先的に効率化していくことでより大きな効果が期待できる。図2に示した再利用パターンからは、変更頻度と規模の大きい(i)利用形態変更を重視した設計を行なうべきであるといえる。以下では、想定される再利用のパターンに対し、どのように変更コストの削減を行なっていくかについて実装方式を踏まえて検討する。

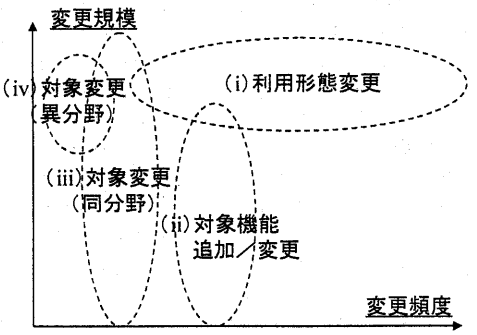

図2:再利用パターン

3.2 シミュレータの機能分割単位の検討

シミュレーション環境の再利用に関しては、従来、どのような単位で再利用をすべきかについて明確な指針が与えられていなかった。ここでは既に指摘したように、シミュレーション環境を「対象製品動作を模擬実行するシミュレータ機能」と「テストを支援するテスト支援機能」という2つのカテゴリの機能集合体としてとらえ、それぞれの特徴に合った形での再利用を考慮した機能分割の指針を与える。

(i)テスト機能の視点からの機能分割

シミュレーション環境で提供するテスト支援機能については、例えば、テストデータの投入、テストの自動実行、テスト結果の記録・提示といった様々な機能サービスが考えられる。再利用の視点からは、これらの機能サービス毎に機能分割し、それらの間のインタフェースを明確にしていく。これによりテスト環境を構成する個々のサービス機能のモジュール化やカプセル化が実現され、個々のサービス機能の再利用性を高めることが可能となる。

この分割方式は、テストの形態やシミュレーション環境の構成が変更されたときに対応しやすく、前述の再利用パターン(i)シミュレーション環境利用形態の変更に相当する。例えば、GUI で実行結果を表示するテスト環境から、テスト結果をファイル出力する環境に変更したい場合、テスト結果出力部を単独のサービスパッケージとして部品化してあれば変更は容易となる。

(ii)シミュレータ機能の視点からの分割

対象ソフトウェアや周辺システムの動作を模擬実行するシミュレータ機能は、模擬実行対象の動作や機能・構造の影響を受けやすい。再利用を考慮した機能分割のポイントとしては、模擬実行対象が提供する機能に着目し、それぞれの機能毎に単独でシミュレーションを行う独立したマイクロシミュレータとして機能構築し、これらの間を明確なインタフェースで接続する方式が有効と考えられる。例えば、携帯電話端末のシミュレーション機能を考えた場合、実際の携帯電話でもっているユーザインタフェース部、プロトコル部、情報端末としてのブラウザ部および外部に相当する外部基地局などをそれぞれ独立したマイクロシミュレータとして用意し、それぞれのシミュレータ間をメッセージ通信などの方式で接続することができる。このような方式を採用した場合、プロトコル仕様の変更などが生じれば、その変更影響はプロトコルシミュレータのみに閉じる形となり、他のマイクロシミュレータの再利用性は保証される形がとれる。このようにこの分割方式は再利用パターンの(ii)シミュレーション対象の機能追加/修正に相当する方式と言える。

シミュレーション環境の再利用では上記の二方式を併用する事でより再利用しやすい構造を作り出すことが可能となるが、その中でも特にどちらの分割を重視するかによって実際の構造が変化する場合がある。一方、分割基準を細かくしすぎると、変更時に差し換えるべきマイクロシミュレータの数が増加し、かえって再利用効率を下げてしまう恐れもあり、これらのトレードオフを考慮して有効な分割単位を使い分けることが求められる。

3.3 シミュレーション環境の実装方式

前述のような指針でシミュレーション環境の機能分割を行った後、実際に環境として実装する場合、選択した実装形態によりその再利用性は大きく変わる。特に実装面での再利用のポイントとしては、今後予想される再利用の頻度を考慮する必要がある。即ち、先の再利用パターンで(i)利用形態の変更や(ii)シミュレーション対象ソフトウェアの機能変更/修正などがどの程度の頻度で発生するかによって、実装の方式を変えていくことが望まれる。

(i) アプリケーションレベルの分割実装

シミュレーション環境を構成する機能（ここではテスト支援機能、シミュレータ機能の両方を含む）をそれぞれ別々のアプリケーションとして分離して実装する方式である。別アプリとして実装された機能を利用する／しないといった選択は、アプリケーションもしくはユーザが実行時に行なう。この場合、面倒なコンパイル作業などを必要とする事無く差し替えが実現できるため、変更頻度の高い場合にはこの方式が有利と考えられる。しかしながら、並行して動作するアプリケーション（機能）が多数ある場合には実行操作が煩雑になったり、実行速度が若干遅くなることが予想され、多数の機能を同時にこの方式で実現する場合には注意が必要となる。

(ii)ソースコードレベルの分割実装

シミュレーション環境で提供する機能を同一アプリケーション上に構築し、ソースコード上の境界によって分離する方式。機能の修正や差し替えが必要になれば、ソースやオブジェクト（モジュール）を差し換えてコンパイル（リンク）することで差し替えを行なう。変更の自由度は高く、プログラムで対応できることならばどのような変更にも対応可能である。しかしながら、変更の度にコンパイルが必要なため、頻繁に変更を行なう場合には効率的ではない。

以上より、提供する機能の組み合わせを実行時に選択するようなシミュレーション環境についてはアプリケーションレベルで分割し、動的な結合が可能で

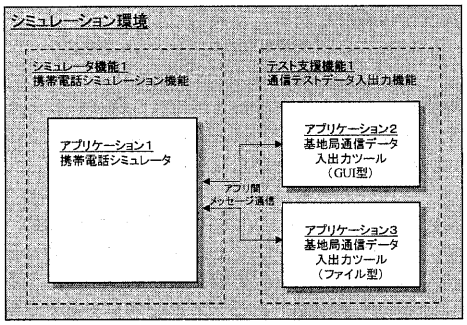

図3:アプリケーションレベルの分割実装

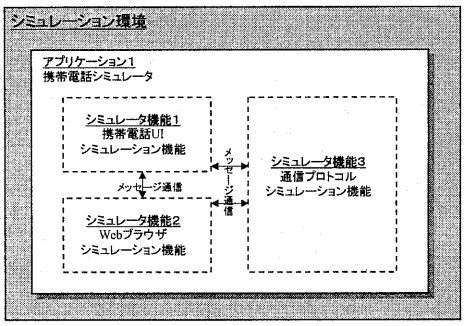

図4:ソースコードレベルの分割実装

あるようにしておくのが効率的であるといえる。逆に上記以外の場合や、高い実行速度を必要とするようなシミュレーション環境には、ソースコードレベルでの分割が適していると考えられる。

3.4 適用事例

ここでは上述方式を実際の携帯電話端末ソフトウェアのシミュレーション環境構築に適用した事例を示す。

[要求されるシミュレーション環境]

携帯電話端末の開発においては、特にユーザインタフェース部の変更が多く、ユーザインタフェース部の制御ソフトウェアをハードウェア無しでテストおよびデバッグするためのシミュレーション実行環境を提供する事が重要になっている。通常、携帯電話端末は約一年に一度のペースでバージョンアップを行なっており、その都度環境の再利用とカスタマイズがもとめられている。今回の適用評価ではユーザインタフェースとして操作キーパネルおよび液晶ディスプレイを持つ携帯電話を対象に従来製品向けに構築されたシミュレーション環境に対し、再利用性の観点からの見直しと実際の再利用を行った。

[再利用パターンの分析]

今回の適用においては、まず、シミュレーション環境に対して、どのような再利用の可能性がありうるかについて再利用パターンの観点から分析した。

(a)対象の機能追加／変更

携帯電話端末ビジネスでは競争の劇化に伴い、様々な機能が次々と追加されるという特色がある。特にユーザの使い勝手の観点から、ユーザインタフェース特に液晶ディスプイなどに関係するハード/ソフトの変更は頻繁に行われる傾向がある。このため、これらを考慮したユーザインタフェース部のシミュレータの変更や改造が必要となるため、これらを想定した設計を行なう必要がある。

(b)利用形態の変更

シミュレーション環境が提供するテスト支援機能についても、これまではワンショットテストの支援が中心となってきたが、対象製品の機能規模増大にともない、多量のテストデータを繰り返し流すといったリグレッションテスト自動化機能も求められるようになっており、これらの機能を状況や目的に応じて使い分けていく必要がある。そのため、テスト形態を実行時に選択できるための設計が必要となる。

[環境のモジュール構成]

上記の再利用パターンの分析結果に基づき、以下のような方針でモジュール構成の設計を行ない、再利用によりシミュレーション環境の再構築を行った。

シミュレーション環境の機能分割単位の検討

(a)の変更は主にハードウェアの変更であり、変更および再利用は仕様変更が発生したハードウェアごとに行われる。このため、携帯電話の各ハードウェアごとに分割し、キーパネルおよび液晶ディスプレイそれぞれの模擬部分をマイクロシミュレータとして構築した。次に(b)の変更対象はテスト実行時のデータ入出力機能であり、これに対応するため、テスト入出力を実装するモジュール集合をテスト支援機能のサブ機能として独立させた。ここでは、手動／自動の二種類を切り替えて使用する必要があるため、それぞれのテスト形態に対応するシミュレータを構築した。

シミュレーション環境の実装方式

シミュレーション環境の実装に関しては、まず、各ハードウェアのマイクロシミュレータについては、ハードウェアの仕様変更は予め想定できないことから、差し替え用のシミュレータは予め用意できない。さらに実行速度を要求されることから、ここではソースコードレベルでの分割方式を採用した。一方、入出力ツールについては変更頻度が多く、また変更内容が予め想定されていることから、アプリケーションレベルでの分割を行った。(図５)

上記のような機能分割および実装方式の採用により今回の適用評価では、携帯電話端末のシミュレーション環境として想定される再利用を吸収可能なシミュレーション環境を構築できた。また対象製品のバージョンアップ時の変更に対して柔軟に対応するとともに、手動／自動といったテスト形態の変更を容

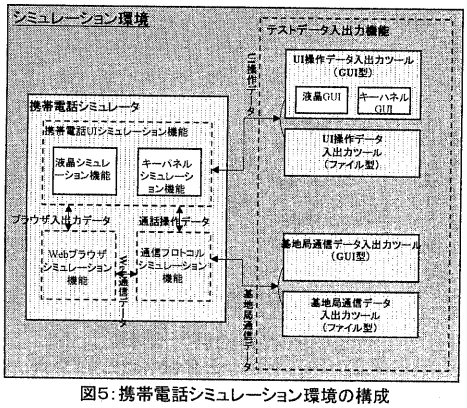

図5：携帯電話シミュレーション環境の構成

易に行なうことが可能となった。

4．シミュレーション環境の信頼性確保

シミュレーション環境を製品ソフトウェアのテストに利用する場合、シミュレーション環境自身の信頼性や品質が製品ソフトウェアのテストの質に大きな影響を及ぼす。特にこれまで述べてきたように本来別の目的や対象向けに構築されたシミュレーション環境を再利用する場合には、その再利用のやり方によって構築されるシミュレーション環境の信頼性は大きく異なってくる。ここではシミュレーション環境再利用時の信頼性確保の視点から、

(i) 再利用の際の品質低下を防ぐための効果的な再利用プロセス

(ii) 再利用により構築されたシミュレーション環境に対する動作確認方式

を提案する。

4.1 効率的な再利用プロセス

4.1.1 全体の流れ

一般にシミュレーション環境を再利用する場合の作業プロセスとしては図 6 に示すような手順を考えることができる。

シミュレーション環境の再利用に際しては、その

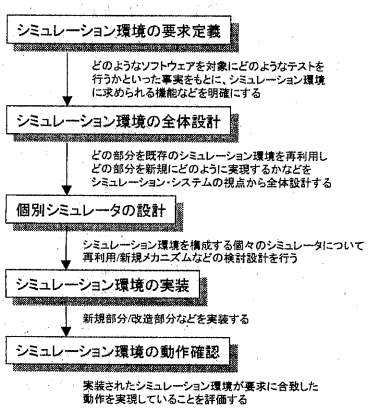

図6 シミュレーション環境の再利用プロセス

品質や信頼性を左右する要因として

・いわゆる「機能の抜け」をいかに防ぐか

・「機能」の誤動作をいかに防ぐか

を大きな課題と考えることができ、図 6 に示した再利用プロセスの中でこれら課題に対する対策を着実に実施していくことが求められる。

4.1.2 シミュレーション機能の実装度/再現度

既に述べたようにシミュレーション環境はその「機能」として「対象製品動作を模擬実行するシミュレータ機能」と「テストなどを支援するテスト支援機能」の 2 つを持ち合わせている。シミュレーション環境の再利用ではこれら 2 つのカテゴリの機能それぞれに対して信頼性確保を視野に入れた再利用性を検討する必要が生ずる。この再利用性の検討について、我々は図 6 に示した再利用プロセスのうちの「シミュレーション環境全体設計」「個別シミュレータの設計」の各段階で、これら機能に関する実装度/再現度の 2 つの尺度を用いた検討を行うことで、より体系的な再利用性の検討を行う方式を考えている。

実装度

シミュレーション環境の再利用を考えた場合、再利用される側のシミュレーション環境が、再利用する側の機能要求をどの程度実現しているかを評価する。

再現度

シミュレーションの精度を考えた場合、再利用される側のシミュレーション環境が再利用する側で求められるシミュレーションの忠実さや精度をどの程度まで実現しうるかを評価する。

実装度/再現度として、それぞれ表 1,2 に示すようなレベル評価基準を用意する。

表 1 シミュレーション機能の実装度

(I-a)	未実装
	再利用する側の要求機能が再利用される側で実装されていない
(I-b)	部分的に実装済み
	再利用する側の要求機能が再利用される側で部分的に実装されている
(I-c)	実装済み
	再利用する側の要求機能が再利用される側で一通り実装されている

これらのシミュレーション機能の実装度/再現度を考慮すると、図 6 の「シミュレーション環境の全体設計」の作業は図 7 のように詳細化できる。

表 2　シミュレーション機能の再現度

(R-a)	**再現不能**
	再利用する側で必要なシミュレーション精度が再利用される側で達成されていない
(R-b)	**部分的に再現可能**
	再利用する側で必要なシミュレーション精度が再利用される側で特定条件では達成されている
(R-c)	**完全に再現可能**
	再利用する側で必要なシミュレーション精度が再利用される側で完全に達成されている

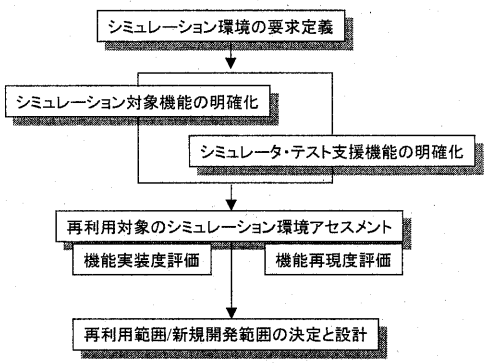

図 7　シミュレーション機能の実現度/再現度評価

4.1.3　適用事例

ここでは先に示した携帯電話シミュレーション環境の再利用を考え、既に先行して利用されていた PDC 向けのシミュレーション環境を利用して新たに cdmaOne™向けのシミュレーション環境を構築するケースをモチーフに事例検証する。

(1)シミュレータ機能の明確化と評価

表 3 に示すように PDC と cdmaOne™では同じ携帯電話であっても機能的に異なるところが多い。例えば電話としてのユーザインタフェース部はほぼ機能的に共通であるが、ベースとなるプロトコル部や周辺にあたる Web ブラウザサービス部などは機能的に大きく異なる。このようなシミュレーション対象となるソフトウェアの機能面の違いを明確にした上で、先に示した機能の実装度/再現度の評価を行った結果を表 3-(b)に示す。これは PDC 向けの既存のシミュレーション環境を cdmaOne™向けに再利用する際の評価結果を示したものであり、UI シミュレータは双方の UI の機能的な差異が小さいことを受けてシミュレーション環境の再利用性も高くなることを示している。一方、プロトコルシミュレータなどについては双方の方式が大きく異なるためほとんど再利用ができないことを示している。

表 3-(a)　シミュレーション対象機能の比較

	PDC	cdmaOne
電話 UI 部	標準 UI	標準 UI
通信プロトコル部	TDMA 方式	N-CDMA 方式
Web ブラウザ部	C ブラウザ	A ブラウザ

表 3-(b) シミュレーション機能の実装度／再現度

	実装度	再現度
UI シミュレータ	(I-c)	(R-c)

表 4-(a)　テスト支援機能の比較

	PDC	cdmaOne
テストデータ投入	手動	自動
テスト実行	手動	自動/手動
テスト動作記録	自動	自動

表 4-(b)　テスト支援機能の実装度

	実装度
テストデータ投入機能	(I-a)
テスト実行機能	(I-b)
テスト動作記録機能	(I-c)

(2)　テスト支援機能の評価

テストツールとしてシミュレータに実装される機能についても同様に実装度/再現度の 2 つの観点から評価する。テストツールとして実現が求められるテスト支援機能としては、表 4 に示すようにテストデータ投入、テスト実行、テスト動作記録などがあげられる。

今回の再利用対象である PDC シミュレーション環境では表 4 -(a)に示すようにテストデータ投入、テスト実行などはマニュアルで対応していたが、cdmaOne™のテスト環境としては、これらについて自動化が強く望まれていた。このため、これらの機能については再利用側と被再利用側との機能の開きが大きく実装度が低い値となっている。結果としてテスト機能の視点からそのまま再利用可能な機能はテスト動作記録のみであり、それ以外は新規および追加実装が必要なことが確認できた。

4.2　シミュレーション環境の動作評価

再利用によりシミュレーション環境を構築する場合、シミュレーション環境が目的に沿った機能を提供しうるどうかを評価する必要がある。特に再利用によるシミュレーション環境では、

E-1　シミュレーション機能の過不足

- 再利用可能な範囲が明らかでないため、本来実装が必要でないシミュレーション機能まで実装する
- 必要なシミュレーション機能が再利用元のシミュレーション環境で欠落しているために、新規構築側の機能が欠落する。

E-2 シミュレーション精度の低下
・機能的には実装されていても、本来期待されるシミュレーションの精度が出せずに結果として不正確なシミュレーションになる

などの事態が想定できる。これらの状況を正確に把握するために、シミュレーション環境内にアサーションを組込みシミュレータ評価動作時に状況を把握する方式を採用する。

(1) 機能過不足のチェック方式

シミュレーション環境は複数のシミュレータや周辺ツールから構成され、これらが相互に呼び合って動作することで系としてのシミュレーション・テスト環境を実現している。機能の過不足をチェックする方式として、これらのシミュレーション環境を構成する個々のツール群のインタフェース部やそれぞれが提供する機能モジュールの呼び出し側および個々の機能モジュールの先頭にアサーションを挿入しておく。これらのアサーションには、個々の機能の名称や起動条件や制約条件などを明示し、シミュレーション環境が動作し、当該機能の起動をかけた時点で、アサーションチェックを行うことが可能となる。　例えば、先の携帯電話のシミュレーション環境について実際に機能レベルのアサーションを挿入した事例では、ユーザインタフェース部のシミュレータがプロトコルシミュレータに情報を渡し起動をかけた時点で、プロトコルシミュレータは、自分が PDC プロトコルを模擬実行するものであることを表明する。一方、ユーザインタフェース側シミュレータは、情報を渡した相手は cdmaOne™プロトコルシミュレータである旨を表明した場合、双方のアサーション間で食い違いがあることが明白になり、その時点でシミュレータ再利用時の機能の過不足を検出することが可能となる。

(2) シミュレーション精度不足のチェック方式

一方、シミュレーションの精度は、機能呼び出し時の入力値に対する出力値の精度、機能呼び出し手順に対する処理結果の精度、機能呼び出しタイミングに対する機能処理時間変化の精度、の３つの観点で評価する。精度不足のチェックも機能不足のチェックと同様に、機能モジュールの呼び出し側および個々の機能モジュールの先頭にアサーションを挿入して実現する。これらのアサーションには、個々の機能で保証する数値精度、操作手順精度、時間精度のレベルを（(R-b)レベルであれば、保証されている内容も）記述し、当該機能の起動時に表明を行なう。ユーザは、動作確認フェーズにおいてこれらを確認し、各機能が要求された精度を満たしていることを確認する。

さらに、シミュレータ操作確認時に操作手順精度あるいは時間精度不足が明らかとなった場合にはログチェックを行ない、不足している内容および程度を確認する。ログチェックでは、シミュレーション機能呼び出しおよび処理時間を逐次記録しておき、シミュレーション後にシミュレーション環境への要求定義と比較して未対応手順や未対応操作タイミングの検出を行なう。例えば、先のプロトコルシミュレータの例では、ユーザインタフェース側がプロトコルシミュレータを起動した時点で要求精度が数値精度完全(R-c)、手順精度完全(R-c)、時間精度なし(R-a)であることを表明する。次にプロトコルシミュレータが実装精度が数値精度完全(R-c)、手順精度部分(R-b)、時間精度部分(R-b)であることを表明して手順精度が不十分なことが判明した場合、引き続きログチェックを行なうこととなる。

5. おわりに

本報告では組込みシステムのシミュレーション環境を効率的に再利用するための設計方式について述べた。再利用を念頭に置いたシミュレーション環境の構成方式として、想定される再利用パターンを考慮し、シミュレーション環境の機能分割と実装時の分割方式を提案した。この方式を採用することで再利用が加速されシミュレーション環境の構築コスト削減に大きな効果をもたらすことが可能となる。

また、より確実な再利用を促進し、再利用における抜けや誤りを最小限に抑えるために、シミュレーション機能の実現度/再現度を考慮したシミュレーション環境の全体設計手順およびアサーションを利用したシミュレーション機能の確認方式などについても検討した。これにより従来課題とされてきたシミュレーション環境自身の信頼性や品質向上などについても効果が得られると考えている。

参考文献

[1] 小山、原、植木．OS シミュレーションによるリアクティブソフトウェアのテスト工程の改善．第 120 回ソフトウェア工学研究会講演集．情報処理学会、1998 年 7 月

[2] 河井、西山．組み込みシステム用ソフトウェア開発環境．情報処理 Vol.5,No.9,pp.872-879、1996 年 9 月．

[3] 中本、高田、田丸．組み込みシステムの現状と動向．情報処理 Vol.38,No.10,pp.871-878、1997 年 7 月

[4] 岡島、五十嵐、小山、安田．組み込みソフト開発支援のためのシステムシミュレーション環境．第 51 回情報処理学会論文集 Vol.5,pp.259-260、情報処理学会、1995 年