ソフトウェア開発で実機レステストを実現する仮想テスト環境

Virtual Testing Environment for Software Development Eliminating Need for Actual Equipment

鷲見　毅 SUMI Takeshi　　小野寺　昭人 ONODERA Akito

ソフトウェア開発では，開発の早い段階からソフトウェアを動作させてテストを行い，不具合を発見・修正して品質を確保する必要がある。しかし，社会インフラシステムなどでは，実際のハードウェアを用いた実機環境でテストを行うことが多く，必要な機材をそろえて実機環境を構築するまでに時間が掛かり，開発スケジュールが遅延するという問題があった。

そこで，東芝グループは，実機環境を仮想化し，実機レステストを可能にする仮想テスト環境を構築した。実機環境と比べ容易に準備できることから，仮想テスト環境を使って早期にテストを開始することで，開発期間の短縮やテスト工数の削減が可能になる。

To assure the quality of software during its development, it is necessary to conduct tests from the upstream stages of the development process by running the software so as to detect any bugs and eliminate their causes. In the social infrastructure field, these tests are often conducted by installing the software in the actual equipment in which it will be used. Considerable time is therefore required to prepare the testing environment, creating a significant issue in terms of delays in the development schedule.

As a solution to this issue, the Toshiba Group has constructed a virtual testing environment that eliminates the need for actual equipment when testing software for social infrastructure systems through the application of virtualization technologies. This virtual testing environment makes it possible to prepare a testing environment more easily compared with the conventional method using actual equipment. It is thus contributing to the shortening of software development periods and the reduction of workloads as it allows testing to be initiated at an early stage.

1. まえがき

ソフトウェア開発を伴う製品開発では，製品出荷後の不具合を防ぐために，出荷までにソフトウェアテストを行い，不具合を修正することが重要である。そのため，開発に占めるソフトウェアテストの比率は大きくなる。IPA（独立行政法人 情報処理推進機構）の調査によると，結合テストと総合テストがソフトウェア開発に占める期間や工数は，開発全体の30％を超える[(1)]。こうした背景から，開発工数を抑え，適切な期間で開発を完了させるには，いかに効率的にソフトウェアテストを行うかが重要になる。テスト期間を短縮し，テスト工数を削減して効率的にテストを行うために，テスト自動化の取り組みが行われている[(2)]。

一方，社会インフラシステムなどの大規模なシステムの開発では，実機でテスト環境を構築し，結合テストや総合テストが行われることが多い。しかし，実機環境を構築するには，実際の運用に用いられる機器一式をそろえなければならない。特に大規模なシステムは，複数のサブシステムが相互に通信しながら動作するため必要な機器も多く，その準備には時間が掛かる。実機環境を用いたテストは，その準備が整うまでは開始できないため，結果としてテストの遅延を招く要因の一つとなっていた。開発期間の30％以上を占める結合テストと総合テストの開始が遅れることは，開発に大きな悪影響を及ぼすため，解決すべき重要な問題といえる。

こうした問題の解決には，実機を用いずに行う実機レステストが有効である。そこで，東芝グループは，実機環境を仮想化し，実機レステストを可能にする仮想テスト環境を構築した。汎用計算機上でテスト対象のシステムを動作させることで，早期に結合テストと総合テストを開始できる。ここでは，実機レステストを行うための仮想テスト環境の構築について，その課題と解決手段を述べる。

2. 実機レステスト実現のアプローチと課題

それぞれのハードウェアで動作しているテスト対象及び周辺機器のソフトウェアを，仮想マシン（VM：Virtual Machine）の上で動作させられるようにする（**図1**）。このように実機環境を仮想化することで，一つの物理的な計算機上でシステム全体を動作させられるようになる。実機レステストを行うためには，こうした仮想テスト環境が必要になる。

実機レステストを行うための仮想テスト環境を構築・運用するには，以下を行う必要がある。

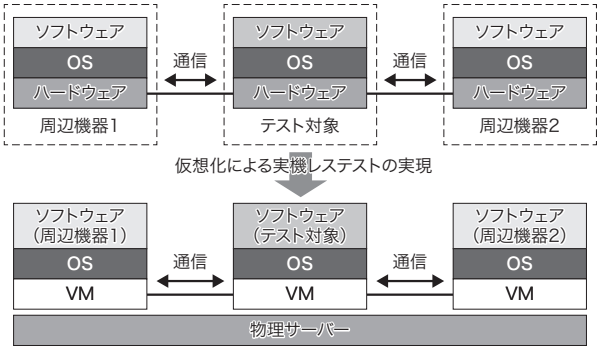

図1. 仮想テスト環境の構築

テスト対象や周辺機器のソフトウェアを，ハードウェアではなくVM上で動作させることで，実機レステストを実現する。

Construction of virtual testing environment using virtual machines as alternative to actual equipment

⑴ テスト対象と周辺機器の仮想化
⑵ 周辺機器の動作模擬の設計・実装
⑶ 仮想テスト環境のセットアップ自動化

まず，⑴に挙げたように，テスト対象と周辺機器をVM上で動作させる必要がある。WindowsやLinux®などのOS（基本ソフトウェア）上で動作し，周辺機器とTCP/IP（Transmission Control Protocol/Internet Protocol）で通信しながら動作するシステムであれば，既存の仮想化ソフトウェアを用いることで，比較的容易に仮想化できる。しかし，各ソフトウェアをVM上で動作させるため，ハードウェアに起因する不具合の発見ができなくなる。そのため，実施するテストがどのような不具合の発見を目的としているかを定め，テストの目的に合致した仮想テスト環境の範囲と仮想化の手段を慎重に検討する必要がある。場合によっては，一部の周辺機器を仮想化せず，仮想テスト環境に実機を混在させて用いるといった判断も必要になる。こうした検討を行わずに仮想テスト環境を構築すると，目的とするテストを十全に行えなくなるおそれがある。

次に，テスト対象と通信する周辺機器に，テストの目的に合致した動作をさせる必要がある。しかし，実機環境では，周辺機器が同じタイミングでエラー応答を返すなど，テストの都合に合わせた動作をしてくれるとは限らない。そのため，周辺機器の動作を模擬する模擬装置（テストモック）を作成し，これを用いてテストが行われることが多いが，テストモックの開発にも工数や期間が掛かる。そこで，⑵に挙げたように，仮想テスト環境で行うテストにとって必要な周辺機器の動作を見定め，テストの目的に合致するように模擬する動作を設計し，実装する必要がある。また，実装に手間取らないように，できるだけシンプルに設計することも重要になる。模擬する動作が複雑であるなどの理由から，実装コストが高くなる場合は，無理にテストモックを作成せず，実際のソフトウェアをそのまま用いる判断も必要になる。

最後の⑶は，仮想テスト環境の構築で必要になる。テスト対象や周辺機器をVM上で動作させるためには，実機環境と同様に，OS設定やネットワーク設定などのセットアップ作業を行う必要がある。しかし，この作業を手動で行うと，仮想テスト環境の準備に多くの時間が掛かる。したがって，仮想テスト環境を必要なときに準備して利用できるようにするために，仮想テスト環境のセットアップ作業の自動化が重要である。

実機レステストをより効率的に行うには，これらを実現して，仮想テスト環境を構築する必要がある。

3. 仮想テスト環境の構築

仮想テスト環境について，放送用システム向けに構築したものを事例に説明する。放送用システムについては文献[3),4)]が詳しいが，ここでは，放送用システム内で番組の収録・送出を行うサブシステムを対象にした。

3.1 テスト対象と周辺機器の仮想化

放送用システムの事例では，テスト対象のソフトウェアの論理的な正しさを確認するテストを，仮想テスト環境で行うことにした。具体的には，次の三つの確認を，仮想テスト環境で行う。

⑴ 周辺機器から制御信号を正しく受信できる。
⑵ 受信した制御信号に対して正しい処理を行える。
⑶ 周辺機器に正しい制御信号を送信できる。

一方で，大量の制御信号を受信するなどの高負荷状態での動作確認や，それらの処理を適切な時間内に完了できるかといった信頼性・性能を確認するテストは，仮想テスト環境ではなく実機環境で行うことにした。

こうしたテストの目的から，テスト対象のソフトウェアが動作することと，制御信号の送受信を行えることの2点を満たす仮想テスト環境を構築すればよいと判断した。放送用システムのソフトウェアは，テスト対象と周辺機器ではともにWindows若しくはLinux®上で動作し，通信もTCP/IPで行われるため，仮想化ソフトウェアを用いることで仮想化できる。そこで，仮想化ソフトウェアの一つであるVirtualBox[5)]を用い，基本的に，テスト対象と周辺機器をそれぞれ一つのVM上で動作させることにした。利用可能な実機が存在している一部の周辺機器については，仮想化して動作を模擬させるより，実機を仮想テスト環境に接続して用いた方が仮想テスト環境の準備に必要な時間を短縮できると判断し，実機を混在させた仮想テスト環境を構築した（**図2**）。同様

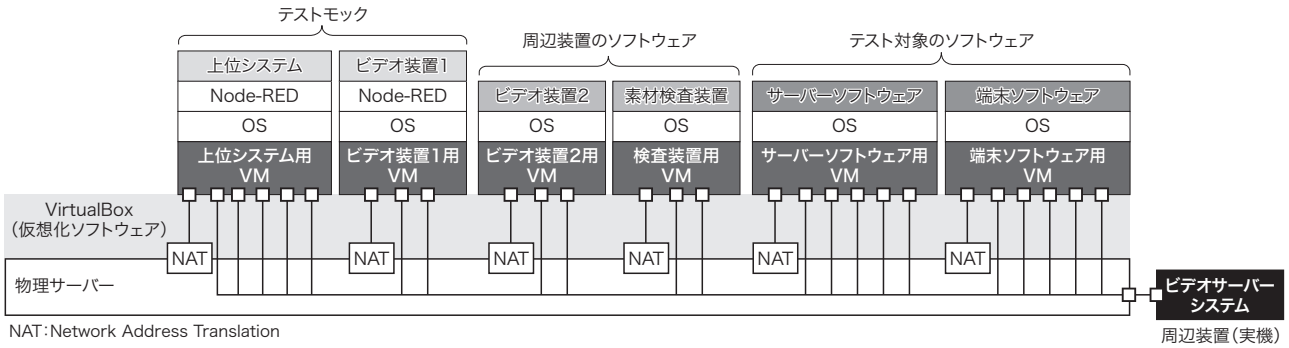

図2. 放送用システムの仮想テスト環境の例

テストの目的に合わせて，VM上で動作させるソフトウェアやテストモックを作る周辺機器を決め，仮想テスト環境を構築した。

Example of virtual testing environment for broadcasting systems

の理由から，動作を模擬せずVM上で実際のソフトウェアを動作させるようにした周辺機器もある。

放送用システムは，制御の種類により制御信号の送受信を行うサブネットに分けられている。そのため，各VMにはサブネットごとにネットワークアダプターを作成し，実機と同様にIPアドレスを設定した（**図3**）。

また，仮想テスト環境に実機を混在させたため，各VMは，VM間だけでなく接続された実機とも通信できるようにネットワーク設定を行う必要があった。こうした外部との通信を可能にする設定も，仮想化ソフトウェアにより提供されており，作成したネットワークアダプターの種類を変更することで実現している。

こうしたネットワーク設定により，VM間及び接続した実機間で制御信号の送受信を可能にした。

3.2 周辺機器の動作模擬の設計・実装

テスト対象を動作させられる最低限の動作が模擬できれば，テストを行うことは可能である。放送用システムの事例では，テスト対象が期待する送信元（IPアドレス）から制御信号が届くことと，期待する宛先（IPアドレス）が制御信号を受け取ることができれば，テストとして必要な動作を行える。そこで，テスト対象との間でやり取りされる制御信号の授受だけを模擬させることにした。すなわち，テスト対象からの制御信号を受信し，受信した制御信号に対する応答を返すことができる簡易的なテストモックを作成した（**図4**）。

作成したテストモックは，TCPなどで送受信される制御信号について，テスト対象から受信した制御信号の内容を解析し，その内容に応じてテストケースで想定されている制御信号を送信する。これにより，テスト対象には，送信した制御信号に対する応答が返ってきたように見える。

こうしたテストモックは，OSS（オープンソースソフトウェア）のフローベースドプログラミングツールであるNode-RED[6]を用いて作成した。放送用システムで作成したような，TCPやHTTP（Hypertext Transfer Protocol）などで制御信号を送信する処理を独自に開発して実装しようとすると，ネット

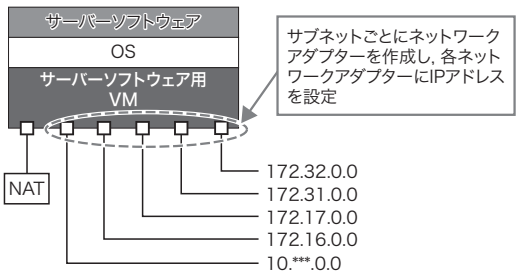

図3. 放送用システムのVMでのネットワーク設定例

VMにネットワークアダプターを複数作成し，複数のネットワークへの接続を再現した。

Example of connections between virtual machine and networks for broadcasting systems

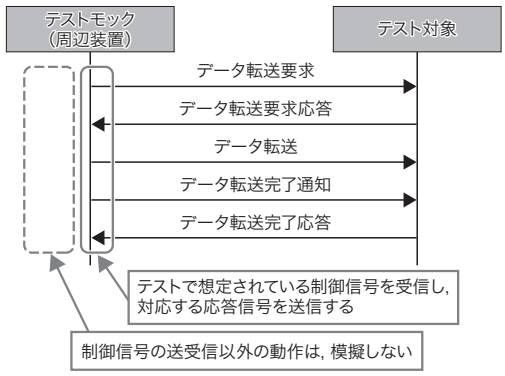

図4. テストモックで模擬する制御信号の範囲

テスト対象との制御信号の送受信だけを模擬し，周辺機器の内部処理は模擬しない。

Simulated control signal ranges of simulator

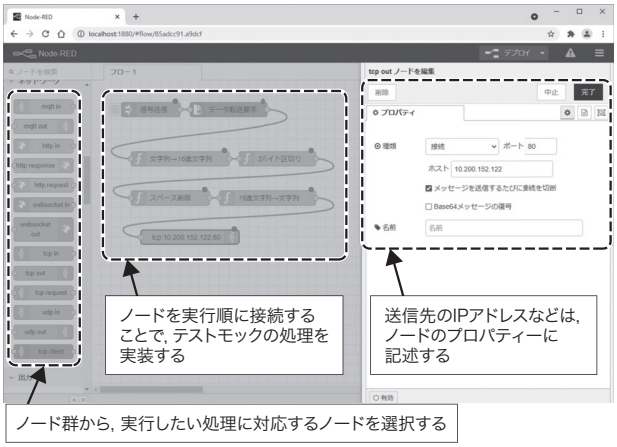

図5. Node-REDによるテストモックへの通信処理ノード実装

エディター上で，実行したい処理がカプセル化されたノードを接続して，テストモックを実装する。

Implementation of communication processing nodes in simulator using Node-RED flow-based development tool

ワークプログラミングの知識が必要になる。しかし，Node-REDには，こうした通信に関する処理が“ノード”という単位にカプセル化されて提供されている。そのため，通信先のIPアドレスや送信するデータを記述するだけで，容易に通信処理が実装できる（**図5**）。Node-REDには，通信処理以外にも様々な処理がノードとして提供されているため，プログラミングに慣れていなくても比較的容易にテストモックを作成できる。

3.3 仮想テスト環境のセットアップ自動化

仮想テスト環境で結合テストと総合テストを行うには，テスト対象やテストモックなどが利用するアプリケーションパッケージや，ライブラリーのインストール，ネットワーク設定などのセットアップ作業を行う必要がある。放送用システムの仮想テスト環境では，Vagrant[(7)]とAnsible[(8)]という二つのプロビジョニングツールを用いて，以下の作業を自動化した。

(1) ネットワークアダプターの作成
(2) IPアドレスの設定
(3) プロキシー設定
(4) ユーザーアカウントの作成
(5) アプリケーションパッケージとライブラリーのインストール

放送用システムの場合，仕向け先に合わせてネットワーク設定などが変更されるため，完全自動化には至っていないが，これらの作業を自動化することで，セットアップ作業に必要な時間を大幅に短縮した。

4. あとがき

実機環境の準備を待たずに，結合テストと総合テストが開始できるように，実機レステストを可能にする仮想テスト環境を構築した。放送用システムの事例では，仮想テスト環境で，テスト対象の論理的な正しさを確認するテストを，前倒しで開始することが可能になった。また，セットアップ作業を自動化することで，容易に仮想テスト環境を準備できるようにした。更に，仮想テスト環境上でテストモックを動作させ，自動テストを実行することも可能になった。

社会インフラシステムでは，TCP/IP以外の通信プロトコルを用いているシステムも多い。今後は，TCP/IP以外の通信プロトコルを用いたシステムでも利用できるように仮想テスト環境を拡張し，社会インフラシステムを中心に展開していく。

文 献

(1) IPA 社会基盤センター．組込みソフトウェア開発データ白書2019．IPA，2019，234p.
(2) 中野隆司，ほか．ソフトウェアのテスト工数・期間を削減するためのシステムテスト自動化技術．東芝レビュー．2018，73，3，p.40–44. <https://www.global.toshiba/content/dam/toshiba/migration/corp/techReviewAssets/tech/review/2018/03/73_03pdf/a10.pdf>，（参照 2021-07-05）．
(3) 東芝．“東芝レビュー 12月号2000 VOL.55 NO.12 特集：BSデジタル放送”．東芝レビュー 55巻12号（2000年12月号）．<https://www.global.toshiba/jp/technology/corporate/review/2000/12.html>，（参照 2021-07-05）．
(4) 東芝．“東芝レビュー 02 2004 VOL.59 NO.2 特集：地上デジタル放送用システム機器設備”．東芝レビュー 59巻2号（2004年2月号）．<https://www.global.toshiba/jp/technology/corporate/review/2004/02.html>，（参照 2021-07-05）．
(5) Oracle. "Welcome to VirtualBox.org!". Oracle VM VirtualBox. <https://www.virtualbox.org/>, (accessed 2021-07-05).
(6) OpenJS Foundation．“概要”．Node-RED日本ユーザ会．<https://nodered.jp/about/>，（参照 2021-07-05）．
(7) HashiCorp. "HashiCorp Vagrant". Vagrant by HashiCorp. <https://www.vagrantup.com/>, (accessed 2021-07-05).
(8) Red HAT. "Move forward faster.". Ansible is Simple IT Automation. <http://www.ansible.com/>, (accessed 2021-07-05).

鷲見 毅 SUMI Takeshi
技術企画部 ソフトウェア技術センター ソフトウェアエンジニアリング技術部
情報処理学会会員
Software Engineering Technology Dept.

小野寺 昭人 ONODERA Akito
東芝インフラシステムズ（株）
府中事業所 放送・ネットワークシステム部
Toshiba Infrastructure Systems & Solutions Corp.