情報処理学会研究報告
IPSJ SIG Technical Report

Vol.2013-MBL-67 No.9
Vol.2013-CDS-8 No.9
2013/9/13

大規模組込み機器向けテスト自動化拡大方式

藤原貴之[†1]　岡本周之[†1]

組込みソフトウェア開発では，増加傾向にあるテスト工数の削減が重要な課題であり，テスト自動化方法が多数提案されている．主な自動化方式には，関数単体のテスト自動化，機器操作の自動化や，ソフトウェアのログ解析による動作判定の自動化がある．こういった従来の手法は外部装置が必要であるか，自動化できるテスト対象が限られていた．しかし，近年のソフトウェアの複雑化により，従来方式では自動化が困難なテストが増加した．また，開発コスト削減の要望から外部装置の利用は導入が難しい場合がある．

本稿では，これらの課題を解決しテスト自動化の適用対象を拡大することを目的として，ソフトウェア内部組み込み型のテスト自動実行エンジンを提案する．本方式では，システムに対する操作とその内部状態取得を抽象化する．さらに，機器操作と操作結果の検証を容易化することによりテストを自動化する．これにより，統合テスト，システムテスト，不具合調査テスト，非再現確認テストの工数を削減することができる．デジタルテレビの開発に本方式を適用した結果，従来のテスト方式と比較して平均 54%の工数を削減できることがわかり，本提案方式の有効性を検証できた．

A Expanded Method for Automated Testing with Large Embedded System

Takayuki Fujiwara[†1]　Chikashi Okamoto[†1]

In embedded software development, reduction in cost for software testing is important issue because the testing cost is constantly increasing. Many methods for automated testing have been proposed including automated unit testing, automated physical device operations, and automated checking of device operations using software log. These previous methods require use of external devices or adapt limited test case However, increased complexity of software system made automated testing more difficult using previous methods. And introduction of external devices are usually difficult because of limited development cost.

In this paper, we propose an automated testing engine incorporated in the target software. This method resolves the issues described above and expands automated test cases. The automated testing engine automates software testing by abstracting operation on the system and its internal state. This makes verification of the system operation and their result easily. Thus, we can reduce testing cost for integral test, system test, bug investigating test, and checking banishment bugs. We applied the proposed method to Digital TV software development and made evaluation of the method. The results of the experiments shown that the proposed method can effectively reduce the testing cost 54% in average compared to previous testing method.

1. はじめに

近年，組込み機器開発では制御機器だけでなく，携帯電話，デジタルテレビなどの情報機器にも適用範囲が広がっている[1]．情報機器では，デジタルコンテンツのレンダリング処理，端末やサーバとのネットワーク処理などが必要になり，それらを処理するためのソフトウェアは大規模化・複雑化が進んでいる[2]．また，経済産業省の実態調査によると，ソフトウェアの規模は 2000 年と 2010 年では 5 倍以上に拡大し数百万行に及んでいる[3]．一方で，情報機器は競合他社の新規参入による価格競争や，販売・流通分野の変化により大型家電量販店が流通の主役を担うようになり価格競争が激化した[4]．その結果，製造メーカは事業課題として開発コストのさらなる削減が求められている．このようなコスト削減を行うため，例えばプロジェクトの約 70%は国内外の協力会社に開発を部分委託し，開発費を削減している[5]．

ソフトウェアの開発規模増大に伴い，開発時に実施されるテスト工程の工数も増加傾向にある．増加するテスト工数を削減する手段として，テスト担当者の手作業で行っていたテストを自動化する方法がある．ソフトウェアのテストには，単体テスト，統合テスト，システムテスト，運用テストがあり[6]，自動化によるテスト工数削減を目的として，これまでにいくつもの方法が考案されてきた．例えば，単体テスト自動化の例として，テスト対象関数のカバレッジを測定する方法があり[7]，統合テスト自動化の例として，ソフトウェアのログを用いて合否判定する方法がある[8]．さらに，システムテスト，運用テスト自動化の例として，PC からの自動操作とカメラを用いた画像認識による結果判定などがある[9]．しかし，ソフトウェアの複雑化により，統合テスト以降ではテストの複雑化・種類の増加が進んでおり，既存の方法では自動化できない統合テストが増えている．また，情報機器開発のシステムテストと運用テスト

†1 株式会社日立製作所
a) 本稿で述べられたシステムおよび製品名は、一般に各社の商標または登録商標である．

の間には，ソフトウェア完成後から出荷までに行う不具合調査テストや，不具合が対策されていることを確認する非再現確認テストがあり，並行して実施されている．これらは長時間の繰り返しが必要で従来のテスト自動化方式が適用可能な例が多いが，開発コスト削減の観点から装置の導入が難しい場合がある．

本稿では，AV・コンシューマ機器などの組込みソフトウェア開発を対象とし，ソフトウェア複雑化と開発コスト削減が求められる中で，従来の方法で自動化が難しいソフトウェアテストを自動化する方式を提案する．提案手法ではソフトウェア内部にテスト自動実行エンジンを埋め込み，ソフトウェアの動作状態を示す変数やログを活用することでテストの自動化を実現する．なお，本稿の提案する手法では統合テスト，システムテストの一部，不具合調査テストおよび非再現確認テストを対象とする．単体テストは自動化方式が多数公開されており，運用テストはソフトウェアの範囲を超えるテストが多いため対象外とした．

2. 従来のテスト自動化方式と課題

本稿で対象としたテストに関する従来の取り組みを，以下に示す．

(1) 統合テスト

統合テストとは，機能ごとに開発した個々のソフトウェア部品内の関数間，または異なるソフトウェア部品同士でのインターフェイス関数間で行うテストであり，広義の意味でソフトウェアの動作状態を判定するテストである．統合テストの対象範囲は広く，テストの対象や方法も様々である．一般的な自動化方式としてソフトウェアの動作ログを解析し動作状態を判定する方法がある[8]．しかし，この方法では決められた文字列のログのみが解析対象であり，拡張性が課題であった．一方，開発対象機器と開発 PC を接続し, PC から開発対象機器のソフトウェアを自動操作して I/0 信号を取得し，I/0 信号からソフトウェア動作を判定するテスト自動化方式が提案されている[10]．しかし，この方法では，統合テストの自動化対象が I/0 に関するものに限定される．したがって，統合テスト自動化対象を拡張する方法が(1)の課題となる．

(2) システムテスト

システムテストは，対象製品全体の動作を確認する．一般的に自動化方式として，ハードウェアの操作を自動化する装置の導入や[9]，装置の出力画像を撮影し，基準画像と比較して判定する方式がある[11][12]．しかし，デジタルテレビのような大規模な機器の場合，画面を撮影するための設備が大掛かりになり，導入コストが増加するだけでなく設置場所の確保が難しくなる．コストと設置場所が(2)の課題となる．

(3) 不具合再現調査テスト

不具合発生報告があったときに直前までの操作を繰り返し実施させ，不具合の発生条件を調べるテストである．

従来は手作業で実施か, PC から機器を操作する仕組みを構築する[13]. しかし，PC から機器を操作する仕組みには準備工数がかかる. また, ソフトウェア開発者だけでなく，テスト担当者も実施するのでソフトウェアに詳しくなくても使える必要がある．さらに，不具合がいつ再現するか不明のため繰り返し実行する必要がある．以上より，テスト担当者でも利用できるテスト自動化方法の構築と，繰り返し実行し，不具合を検出する仕組みの構築が課題となる．

(4) 非再現確認テスト

非再現確認テストは，不具合を修正した後に不具合発生条件となった操作を繰り返し実行し，再現しないことを確認するテストである．従来の方式と問題点は(3)と同等であるため省略する．

3. テスト自動化対象を拡大するテスト自動化方式

2 章で示した課題を解決する技術を検討した．

(1)統合テスト

テストの自動化対象を拡張するために，ソフトウェアの動作状態を示す変数や，デバッグ用のログを利用する．これらは多様な状態を示すので統合テストの対象範囲の自動化拡張に対応できる．

(2) システムテスト

外部装置の設置コストと場所の確保の課題を解決するために，テスト自動化方式を対象機器の外部ではなく内部に実装する．内部に実装することで，対象機器分の外部装置を準備するコストと新たな設置スペースが不要になる．

(3) 不具合再現調査テスト

テスト担当者でも利用できるテスト自動化方式を構築するために，スクリプトコマンドを用いてテストを実行させる．スクリプトコマンドの使用によりソースコードの修正を不要とすることで，テスト担当者でもテストの自動実行が容易になる．また，長時間実施可能とするために，テストの繰り返し実行と不具合検出機能を搭載する．繰り返し実行機能と不具合検出機能により長時間実施が必要な試験の自動化が可能となる．

(4) 非再現確認テスト

解決技術は(3)と同一である．

すなわち，本稿で提案するテスト自動化方式の要件は以下の通りである．

(1) ソフトウェアの動作状態を示す変数やログの活用
(2) テスト自動実行エンジンをソフトウェア内部へ実装
(3) スクリプトコマンドによるテストを実行
(4) 繰り返し実行機能を搭載

3.1 ソフトウェアの動作状態取得および判定機能

組込み機器ソフトウェアには状態遷移が多数含まれている．一般に確認すべきソフトウェア中の状態は，単純な変数，状態管理テーブル，その他複雑な状態に分類される．本節では，3 章の要件(1)に対応し，それぞれの状態を取得し判定するための方式を検討した．

3.1.1 動作結果通知関数挿入による単純な変数取得機能

本提案方式では，まず動作結果を通知する関数の呼び出しを対象ソフトウェアに追記する．次にテスト対象関数が動作結果通知関数を同期呼び出して変数の値を取得することで状態を取得する．図 1 に動作結果通知関数挿入によるテスト自動実行シーケンスを示す．

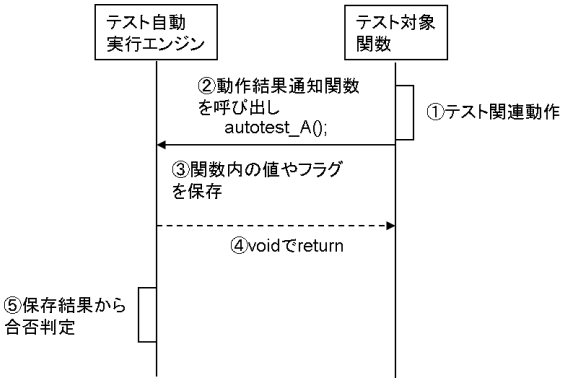

図 1 動作結果通知関数挿入による
テスト自動実行シーケンス

Fig. 1 Automated test sequence with insertion of notification function

図 1 では，テスト対象関数の中に動作結果通知関数である”autotest_A()”が挿入されている．”autotest_A()”はテスト対象関数の値や戻り値を通知したり，関数の動作を示すフラグを変更する．①まず，テスト対象関数がテストに相当する動作を行う．②次にテスト対象関数が”autotest_A()”を同期呼び出しする．③”autotest_A”はテスト対象関数の状態を示す変数やフラグを取得し，テスト自動実行エンジンが確認可能な場所に保存し，④void で return し終了する．⑤テスト自動実行エンジンは保存した状態を示す変数とテスト合格基準となる値を比較し，合否を判定する．しかし，本方式はテスト対象ソフトウェアへ修正を加えるため，使用不可避な場所に限定して使うのが望ましい．

3.1.2 状態管理テーブルの確認機能

組込み機器では多数のアプリケーションが起動し，ミドルウェア層が状態管理テーブルを持ちこれらの状態を制御することで製品としての機能を実現する．従来方式では状態管理テーブルの値をログに出力していたため，テスト用のログを埋め込む作業が必要であった．本提案方式では，状態管理テーブルと連携し，アプリケーションの状態を判定する仕組みを内部に構築することでテストを自動化する．図 2 に，状態管理テーブルと連携した判定シーケンスを示す．図 2 では，①まずテスト自動実行エンジンが状態管理テーブルを参照し，②テスト対象のソフトウェアモジュールの状態をポインタで取得する．

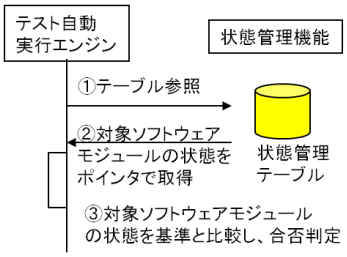

図 2 状態管理テーブルと連携した判定シーケンス

Fig. 2 Judgment sequence with state management table

③続けて，取得したポインタをテスト合格基準となる変数と比較し，対象ソフトウェアの合否を判定する．

3.1.3 複雑な状態取得機能

状態取得手段の一つに，ログ出力用のコードに状態を示す表記を埋め込み，出力されたログを確認してテストの合否を判定する方法がある[8]．一方，組込み機器ではデバッグ用にソースコードにログ出力用のコードを埋め込み，オプションなどでターミナルソフトへの出力量を制御する機能を導入することが多い．本提案方式では組込み機器のログ制御機能と連携し，ターミナルソフトに出力するログと，テスト判定基準となるログを比較できる仕組みを構築する．これにより複雑な状態であってもログの出力結果によりテストが可能となる．既存のログ出力量制御機能と連携したテスト実行エンジンの構成を図 3 に示す．

図 3 ではテスト用 PC とテスト対象の組込み機器をシリアルケーブルで接続してあり，テスト対象組込み機器にはログ制御機能が入っている．ログの出力と自動判定までの手順は次の通りである．①ソフトウェア内部のアプリケーションやミドルウェアはログ制御機能を利用してログを出力する．②ログ制御機能には修正が加えられており，①のログをテスト実行エンジン用のリングバッファにコピーする．③テスト実行エンジンはリングバッファの内容と基準の文字列を比較し，比較結果を出力する．

3.2 ソフトウェア内部埋め込み型のテスト自動実行エンジン

3 章の要件(2)で述べた通り，外部装置の設置ではなく，対象ソフトウェア内部にテスト自動化機能を埋め込む方式を検討する．外部装置の機能としては，機器の自動操作，テスト結果の自動判定，テスト結果ファイルの出力がある．本稿で対象とするテスト自動実行エンジンは，上記の機能を代替できるように設計する．

(1) 機器の自動操作

対象機器の操作を受け付けるソフトウェアドライバに操作指示を出せるようにテスト実行エンジンを設計する．

従来の操作手段について，例えば携帯電話であれば利用者の使うテンキーなどが該当する．従来はソフトウェアのドライバ，操作管理機能を経由して対象の操作が可能であるが，本稿ではテスト自動実行エンジンが操作管理機能にアクセスすることで対象機能の自動操作を可能とする．

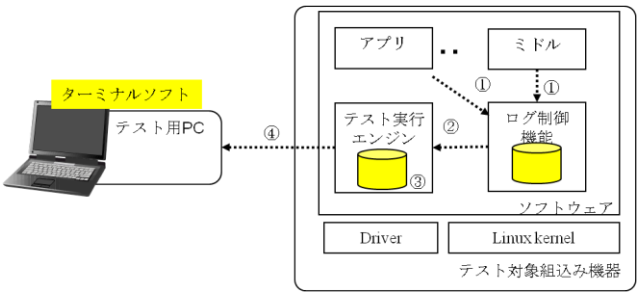

図 3 既存のログ制御機能と連携した
テスト自動実行エンジンの構成

Fig. 3 Architecture of automated testing engine that cooperate with log management system

(2) テスト結果の自動判定

3.1 節に記載の技術をテスト自動実行エンジンに搭載することで実現する．

(3) テスト結果の出力

結果確認が容易になるような形式で出力する．出力形式の例にはテキスト，HTML，XML などがある．

3.3 スクリプトコマンドによるテスト実行

本節では，3 章の要件(3)に対応し，スクリプトコマンドからテストを実行する方法について検討する．スクリプトコマンドからテストを実行する場合，スクリプトコマンドの解析，スクリプトコマンドを用いたテスト対象組込み機器の操作・テストの判定指示，結果出力指示をテスト自動実行エンジンが備えることが必要である．そのため、テスト自動実行エンジンは，各スクリプトコマンドに対応した Application Program Interface(API)を持ち，操作 API によりテスト対象組込み機器を操作し，収集 API によりテストに必要なログや変数を取得する．

3.4 繰り返し実行機能

3 章の要件(4)に対応する．3.2 節で述べたテスト自動実行エンジンに繰り返し実行機能を搭載し，特定のスクリプトコマンドの記述により指定したスクリプトファイルを繰り返し実行できるようにする．

4. デジタル TV 向けテスト自動実行環境の開発と適用検討

3 章で示したテスト自動化方式に従い，デジタル TV 製品のソフトウェア開発用にテスト自動実行環境を構築した．

4.1 デジタル TV 向けテスト自動実行環境の概要

デジタル TV 向けのテスト自動実行環境の構成を図 4 に示す．3.2 節で示したように本手法では，デジタル TV ソフトウェア内部にテスト自動実行エンジンを組み込んでいる．また，3.3 節で示したようにテスト自動実行エンジンは各スクリプトコマンドに対応した API を持つ．テスト自動実行エンジンはスクリプトファイルを読み取り，ファイルに記載されているスクリプトコマンドに対応する API を呼ぶ．

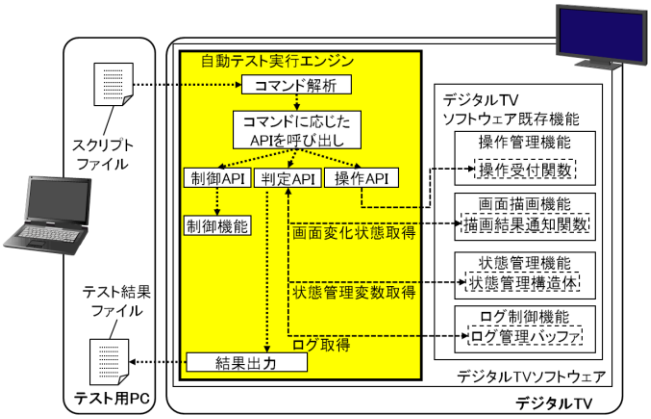

図 4 デジタル TV 向けテスト自動実行環境の構成

Fig. 4 Automated testing environment for Digital TV

API にはデジタル TV を操作する操作 API，テストに必要なログや変数を収集し判定する判定 API，テスト自動実行エンジンを制御する制御 API がある．各 API はデジタル TV ソフトウェアの既存機能を呼ぶが，既存機能にはそれぞれ修正が追加されており点線の四角で表されている．

操作 API は 3.2 節(1)の方式を採用し，デジタル TV の操作を実現する．デジタル TV ソフトウェアの操作管理機能には操作 API が呼び出し可能なインターフェイス関数として操作受付関数が追加されている．

判定 API について，デジタル TV ソフトウェアの画面描画機能には，3.1.1 項の方式を採用し，描画結果通知関数が追加されている．状態管理機能には 3.1.2 項の方式を採用し，状態管理構造体が追加されている．ログ制御機能には，3.1.3 項の方式を採用し，ログ管理バッファが追加されている．これらを用いてテスト自動実行エンジンは判定に必要なログや変数を取得し，合否を判定する．

制御 API は 3.4 節の方式など，テスト自動実行エンジンそのものの制御を行う．スクリプトコマンドの数だけ上記を繰り返すと，結果をテキスト形式のファイルにしてテスト用 PC に返す．

ここで，テスト自動実行エンジンが解析するスクリプト言語の候補としては，一般的なスクリプト言語の Perl, Ruby, Python などがあり，たとえば Ruby を活用したテスト自動実行方法が提案されている[14].しかし，本提案方式で対象としたデジタル TV は ROM 容量が極めて限られており，それらの実行用ライブラリを組み込むだけの余裕がない．

そのためデジタル TV ソフトウェアのライブラリを流用する方針とした．組込み機器開発では開発対象の 90%が C 言語で開発されており[15]，デジタル TV ソフトウェアも C 言語ベースである．そこで，必要なテスト実行機能に絞った軽量なテスト自動実行エンジンを C 言語ベースで開発した．

4.2 デジタル TV の自動操作機構

テスト自動実行を実現するため，デジタル TV の赤外線リモコンの信号に応じた操作を自動化した．
図 5 に, デジタル TV 向けのリモコン自動操作機構を示す．

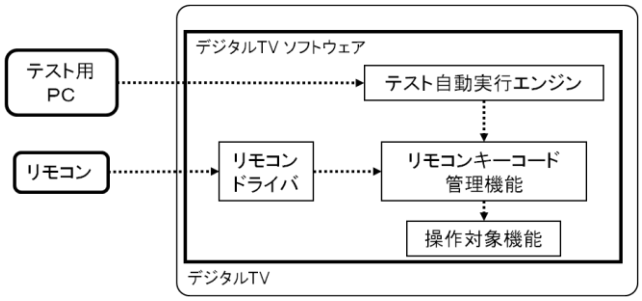

図 5 デジタル TV 向けのリモコン自動操作機構
Fig. 5 Automated controlling of remote controller for Digital TV

製品本来の動きでは，リモコンのボタンが押されると，ハードウェアの赤外線受光部がリモコンからの信号を受けて，デジタル TV のリモコンドライバが動作し，リモコンキーコード管理機能を通じて，リモコンの指示に応じた操作担当機能に処理を振り分ける．本方式では，テスト用 PC がデジタル TV の操作指示を出すと，テスト実行エンジンがリモコンキーコード管理機能に処理を渡すことでデジタル TV の操作を実現している．

4.3 スクリプトファイルの例

3.3 節で示した方式に従い, デジタル TV 向けに作成したスクリプトコマンドの一覧を表 1 に示す．

表 1 スクリプトコマンドの一覧
Table. 1 List of script commands

#	コマンド名称	機能	種類
1	hit	引数のキーコードに応じてデジタル TV に操作コマンドを送信	操作
2	wait	指定秒数だけ実行停止	制御
3	echo	結果ファイルにコメント出力	制御
4	setRepoFile	テスト結果ファイルの出力先指定	制御
5	ckAppBoot	指定アプリの起動状態を確認	判定
6	ckLogSet/ckLogStart/ckLogStop	検索対象文字列を指定し，検索開始と終了を指示	判定
7	waitUntilCkLog	ckLogSet で指定された文字列の出力か引数の秒数だけ実行停止	制御
8	ifLastCkLogNeq	ckLogSet で指定した文字列が出力されなかったら真	制御
9	ckDispStart/ckDispStop	画面変化を確認	判定
10	runLoop	指定したスクリプトファイルを繰り返し実行	制御

表 1 に示すスクリプトコマンドは，4.1 節で述べたようにテスト自動実行エンジンの持つ API を呼び出すことで，デジタル TV の操作，テスト結果ファイルの出力指示，テスト対象アプリの状態確認，文字列検索，画面変化確認，繰り返し実行指示などが可能になる．

図 6 はアプリケーションの起動確認と画面の無変化確認を行うスクリプトファイルの例である．”#”が付いた行はコメントと解釈され，日本語も記述可能である．”setRepoFile”によりテスト結果ファイルの出力場所を指定する．”echo”コマンドによりレポートファイルにコメントを挿入する．”hit”コマンドにより引数のキーコードに応じてデジタルテレビを操作する．ここでは”KEYCODE_EPG”とあるため番組表(EPG)を起動する．

```
## 操作，起動確認，および
画面変化確認の例
## レポートファイル作成
setRepoFile
./autotest/report.txt

## [TEST] 番組表確認
echo "--------------"
echo "Now Check EPG"
echo "-------------"
hit KEYCODE_EPG
wait 2
ckAppBoot EPGTASK ON
hit KEYCODE_EPG
wait 2
ckAppBoot EPGTASK OFF

## [TEST] 画面無変化確認
echo "--------------"
echo "Now Check Disp"
echo "-------------"
ckDispStart
hit KEYCODE_YELLOW
ckDispStop
```

図 6 番組表起動確認を行うスクリプトファイル
Fig. 6 Test script file for checking start up of EPG

続いて”wait”コマンドにより，引数で指定した数値の秒数だけスクリプトファイルの実行を停止する．”ckAppBoot”コマンドは 3.1.2 項の方式を採用しており，引数で指定したアプリケーションの起動状態を確認する．例えばEPGTASK，ON の場合，番組表が起動していればテスト成功であり，起動していなければテスト失敗となる．

次に，”ckDispStart”で画面変化確認を開始する．”ckDispStart”,”ckDispStop”は 3.1.1 項の方式を採用している．”ckDispStop”コマンドが呼ばれるまでに画面変化があるとテスト結果ファイルに記載される．ここでは EPG を OFF した後はリモコンの黄色ボタンを押しても仕様上画面変化は起きないため，テストが成功する場合テスト結果には何も記載されない．

図 7 は出力ログと基準文字列を比較するスクリプトファイルとテスト結果ファイルの例である．”ckLogSet”, “ckLogStart”, “ckLogStop”コマンドは 3.1.3 項の方式を採用している．”ckLogSet”コマンドではテストの合格基準となる文字列を指定する．指定する文字列には正規表現の利用が可能である．次に”ckLogStart”でログ検索を開始する．ここでは 3 秒以内に見つかることを期待しており，3 秒後に終了する．テスト結果ファイルでは，各コマンドの成否が SUCCESS か FAILURE で表示される．また，文字列が見つかると 5 行目のように”[TASK_A] sampleTest result = 0”という文字列が見つかったことを示す．

```
##出力ログと基準文字列との比較例
## 選局(地デジ011)
hit KEYCODE_DIGITAL_011

# ログ取得設定
ckLogSet " [TASK_A]* result = 0" SUCCESS
#ログ取得開始
ckLogStart
wait 3
#ログ取得終了
ckLogStop
```

```
#テスト結果ファイルの例
SUCCESS      :ckLogSet[[TASK_A]* result = 0][SUCCESS]
SUCCESS     :ckLogStart
SUCCESS      :wait[3]
** SUCCESS ** detects [[[TASK_A]* result = 0] from
[TASK_A] sampleTest result = 0]
SUCCESS      :ckLogStop
```

図 7 指定したログ文字列を検出するスクリプトファイルとテスト結果ファイル

Fig. 7 Test script file for matching log character and its result file

図 8 は選局を行った結果，特定のログ出力有無を繰り返し自動判定するテスト向けのスクリプトファイルである．runLoop コマンドは 3.4 節の方式を採用しており ckChannel.aut を無限回(0 は無限と設定)実行させる．ckChannel.aut では，”setRepoFile”コマンドでレポートファイルの出力先を指定し，”ckLogSet”コマンドで基準文字列を指定し，”ckLogStart”コマンドで検索を開始し，”hit”コマンドでデジタル TV を操作した後”waitUntilCkLog”コマンドで最大 3 秒待つ．ここで 3 秒以内に文字列を見つけられればすぐに次のコマンドが実行される．”ifLastCkLogNeq”コマンドは条件分岐を行い，ログが検出されなかった場合に限り”echo”コマンドでテスト結果ファイルにエラーメッセージを出力する．”ckLogStop”で検索終了後，別のチャンネルに選局してからスクリプトファイルが終了する．以降，”runLoop”コマンドにより上記の一連の操作が繰り返し実行される．

```
#ckChannel.autを繰り返し呼び出し
runLoop ckChannel.aut 0
```

```
# ckChannel.aut #
#選局による指定文字列の出力判定テスト
#レポートファイルを設定
setRepoFile./autotest/report_testcase001.txt
#基準文字列を設定
ckLogSet "<ADR:0001:0001>" SUCCESS
#検索開始
ckLogStart
#操作
hit KEYCODE_DIGITAL_011
#検出するまで最大3秒待つ。検出したら次のコマンドへ進む
waitUntilCkLog 3
#条件分岐．ログを検出できなかったら真．
ifLastCkLogNeq SUCCESS
#レポートファイルに出力
echo "***************"
echo "FAILURE on DIGITAL 011"
echo "***************"
endif
#検索終了
ckLogStop
#別のチャンネルに選局
hit KEYCODE_DIGITAL_012
```

図 8 繰り返し実行するテストスクリプトファイル

Fig. 8 Test script file for continuous testing

従来は手作業でリモコンを操作し，画面変化を目視確認してテストを繰り返し実施していた．この場合数十インチの画面変化を連続確認することでテスト担当者の目が疲れ，テスト結果の確認が効率的ではないという問題があったが，本方式により状態判定が効率的にできるようになった．

5. デジタル TV 開発への適用評価

4 章で開発したデジタル TV 向けテスト自動化環境を，実際のデジタル TV 製品のソフトウェア開発に適用し，評価した．

5.1 製品開発での適用対象

デジタル TV のソフトウェア開発における結合テスト，システムテストの一部，再現確認試験，非再現確認テストの一部にテスト自動化環境を適用した．

5.2 適用手順

(1) 結合テスト

デジタル TV ソフトウェアの一部について，3.1.3 項と 3.1.2 項の方法を用いて状態確認のテストを自動化した．

適用時は，表 1 に記載の”hit”, “ckAppBoot”, “cKLogSet”, “ckLogStart”, “ckLogStop”コマンドおよび各種制御コマンドを用いて操作と判定を行うスクリプトファイルを作成し，テストを自動実行した．テスト対象としたソフトウェアモジュールの規模は 55KStep であり，適用したテスト件数は 915 件である．

(2) リモコン操作無反応確認テスト

リモコンの入力により仕様通り反応しないことを確認するリモコン操作無反応確認テストに適用した．リモコン操作無反応確認テストはシステムテストの一種であり，従来テスト担当者のリモコン操作と目視確認で実施されていた．しかし，反応しないことを目視確認するのは難しく，従来の方法では多くの時間がかかっていた．適用時は，表 1 に記載の”hit”, “ckDisp”および各種制御コマンドを用いて操作と判定を行うスクリプトファイルを作成し，テストを自動実行した．適用したテスト件数は 3000 件である．

(3) 不具合再現調査

番組録画や再生の異常終了やネットワークを用いた操作などの不具合発見時，発見者の情報を元に，表 1 に記載の”hit”, “runLoop”および各種制御コマンドを用いて直前の操作に相当するスクリプトファイルを作成し，繰り返し実行した．適用したテスト件数は 14 件である．

(4) 非再現確認テスト

(3)の不具合修正後に，同様のスクリプトファイルを繰り返し実行した．適用したテスト件数は 8 件である．

5.3 評価項目・評価方法

テスト自動実行環境の適用評価について，評価項目と評価方法を以下に示す．

(1) 準備工数

テスト自動実行環境の導入，使用方法理解工数，およびスクリプトファイルの作成工数を算出した．

(2) テスト工数削減率

前年度の開発実績を参考にし，本方式の適用有無において工数削減率を算出した．

5.4 評価結果

5.4.1 準備工数

準備工数を表 2 に示す．不具合再現調査と非再現確認テストは，同一の準備で対応できるため一つにした．環境構築において，テスト自動実行環境は開発者・テスト担当者の使う PC に導入した．また，事前に利用マニュアルを作成し，不明な点は直接答えるようにして早期理解を促進させた．スクリプト作成工数において，リモコン操作無反応確認テストは仕様書からスクリプトファイルを作成するツールを利用して作成した．

表 2 準備工数

Table. 2 Preparation cost

#	対象テスト	環境構築(人日)	スクリプト作成(人日)
(1)	結合テスト	2.0	5.0
(2)	キー受付無反応確認テスト	0.1	0.1
(3)(4)	不具合再現調査/非再現確認テスト	4	2.6

適用マニュアルと現場での適用支援もあり，のべ数十人の利用の割に適用工数は 10 数人日程度と少なく抑えることができた．今回は利用者がスクリプトファイル作成の自動化を行ったことも影響し，スクリプトファイル作成工数も抑えられたが，今後はスクリプトファイル作成の自動化や簡易化対応が課題である．

5.4.2 テスト工数削減率

テスト自動実行環境適用によるテスト工数削減率は，本方式未使用時の工数を C1，本方式適用時の工数を C2，本方式準備工数を C3 とすると， {C1 – (C2+C3)}/C1 で表される．C1 については，前年度までのデジタル TV の開発実績から予測して算出した．テスト削減率を表 3 に示す．

表 3 テスト工数削減率

Table. 3 Reduction rate of testing cost

#	対象テスト	C1:従来作業での工数(人日)	C2:本方式適用時の工数(人日)	C3:本方式の準備工数(人日)	テスト削減率(%)
(1)	統合・結合テスト	23	16	7	0
(2)	キー受付無反応確認テスト	5.6	2.8	0.2	43
(3)	不具合再現調査	32	0.05	6.6	77.8
(4)	非再現確認テスト	54.7	0.05	0	99

本稿では，本方式適用による工数削減率を調べるために適用対象を限定した．その結果(1)については削減工数と準備工数が差し引きで 0 となったが，適用対象を増やし，別の派生機種に本方式を展開していくことで工数削減に貢献できると考えられる．(3)と(4)は，従来の方式で実施する場合，多量の時間がかかりテスト担当者の負担になっていた．本方式の適用によりテスト実施が簡易化されるため，テスト担当者の負荷を減らすことが期待できる．

また，テスト自動実行環境の準備工数を考慮したときの工数削減率を平均すると，(0+43+77+99)/4 = 54.5 %の削減率となる．継続的に本方式を適用していくことで従来自動

化の難しかったテスト工程について，平均して 54.5%の工数削減に寄与できる．

6. 関連研究

本論文の対象は組込み機器の統合テスト，システムテスト，一部の自動化である．

テスト自動化の文脈では，単体テスト自動化の研究が多数行われている．単体テストは，テスト対象となる関数内の全ての分岐が実行されるように引数に値を入力し，戻り値が詳細仕様通りであることを確認するテストである[6]．単体テストの自動化方法としては，テスト自動実行用のソースコードを自動生成する方法[16][17]や，Sulu 言語を用いたテスト自動実行方法[18]などが提案されている．テストケースの自動生成はテストの自動化に有用だが，本稿では複数のテスト工程のテストを対象としており単体テスト向けのテストケース自動生成方式をそのまま使うことは難しい．また，Sulu 言語は単体テスト向けに Java 言語をベースとして設計されたプログラミング言語である．各テストにつき Sulu 言語のテスト用コードをテスト対象ソフトウェアに追記する必要がある．本稿で提案する技術では，そのような必要なくテストの自動化が可能となる．

次に，非組込み機器のテスト自動化技術について検証する．例えばWebサービスやWebアプリケーション開発では，テストの自動化が進んでいる．例えば，JavaScript ベースの Web アプリケーションの受け入れテストを自動化する方式[19]や，開発済みの箇所から不具合が発生していないかを確認する回帰テストを自動実行する方式がある[20]．これらの方法はテスト工数削減に有効だが，Web ベースのアプリケーションでは実機が不要であり，ネットワーク経由でのアプリケーション制御や判定ができる仕組みが構築しやすい．一方，組込み機器はテスト実施に実機が必要であり，機器の利用環境が限定されているなど，固有の状況があり Web アプリケーションの方式をそのまま使うことは難しい．本稿ではデジタル TV での適用を検証したが，本方式はC言語ベースであれば他の組込み機器への展開も可能であり，適用可能なテスト工程も広いことから，幅広い機器，テスト工程に適用できると考える．

7. 終わりに

テスト工数の削減を目的に，テスト自動化拡大方式を提案し，ソフトウェア内部組み込み型のテスト自動実行技術を提案し，結合テスト，リモコン操作無反応確認テスト，不具合再現調査,非再現確認テストに適用した．この結果，それぞれでテスト工数を削減し，平均 54.5%の工数削減率であることを確認した．今後はテスト実施工数を削減し，テスト自動化を促進するために，スクリプトファイルの作成自動化が課題である．

参考文献

1 中島達夫: 画像処理技術者のための組込み入門，映像情報メディア学会誌, Vol. 63, No. 5, PP.633-637(2009).
2 経済産業省: 情報家電産業の競争力を支えるソフトウェア,P.2(オンライン)，入手先 (http://www.rieti.go.jp/it/elife/docs/elife-chapt6.pdf) (2004).
3 経済産業省: 組込みシステム産業の課題と政策展開, P.13(オンライン),入手先 (http://www.jasa.or.jp/et/ET2011/visitor/images/pdf/S1_web_data111116.pdf) (2011).
4 経済産業省: 情報家電産業の収益力強化に向けた道筋,P.7(オンライン)，入手先 (http://www.rieti.go.jp/it/elife/docs/elife-souron.pdf) (2004).
5 経済産業省: 組込みソフトウェア産業実態調査, P.42(オンライン),入手先 (http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2010software_research/10keiei_houkokusyo.pdf) (2010).
6独立行政法人 情報処理推進機構: ソフトウェアテスト見積もりガイドブック, P.191-202(オンライン)，入手先 (http://www.ipa.go.jp/files/000005132.pdf) (2008).
7 Juei, C., Richardson and D. J., : ADLscope: an automated specification-based unit testing tool, *Proc.IEEE International Conference on Automated Software Engineering*, P.289-292(1998).
8 Andrews, J.H. : Testing using log file analysis : tools, methods, and issues, *Proc, IEEE International Conference on Automated Software Engineering*, P.157-166(1998).
9 Pekovic, V., Teslic, N., Resetar, I., and Tekcan, T. : Test management and test execution system for automated verification of digital television systems, *Proc. IEEE Internatioal Conference on Symposium on Consumer Electronics*, P.1-6(2010).
10 Dae-Hyun, K., Joonwoo, S., Seon-bong, L. and Ivan, W.: Automated Testing for Automotive Embedded Systems, SICE-ICASE, P.4414-4418(2006).
11 Kastelan,I., Katona, M., Pekovic, V. aqnd Mihic, V. : Automated functional verification of digital television systems using camera, *Proc. ELMAR*, P.69-72(2010).
12 Borjesson, E. and Feldt, R. : Automated System Testing using Visual GUI Testing Tools: A Comparative Study in Industry, *Proc. IEEE International Conference on Software Testing, Verification and Validation*, P.350-359(2012).
13 Kastelan, I., Pekovic, V., Teslic, N., Tekcan, T., and Marijan, D.: Automatic Black Box Testing of television systems on the final production line, *Proc. IEEE International Conference on Consumer Electronics*, P.869-870(2011).
14 Mate, S., Kai, Qian. and Xiang, Fu. : The automated web application testing (AWAT) system, Education Technology and Training. and International Workshop on Geoscience and Remote Sensing, P.88-93(2008)
15 経済産業省: 組込みソフトウェア産業実態調査, P.23(オンライン)，入手先 (http://www.meti.go.jp/policy/mono_info_service/joho/downloadfiles/2010software_research/10project_houkokusyo.pdf) (2010).
16 Fix, G.: The Design of an Automated Unit Test Code Generation System, International Conference on Information Technology: New Generations, P.743-747(2009).
17 Wijayasiriwardhane, T. K., Wijayarathna,P. G. and Karunarathna, D. D: An automated tool to generate test cases for performing basis path testing, International Conference on Advances in ICT for Emerging Regions(ICTer), P.95-101(2011).
18 Tan, R.P., and Edwards. S. : Evaluating Automated Unit Testing in Sulu, Software Testing, Verification, and Validation, International Conference on Software Testing, Verification, and Validation, P.62-71(2008).
19 Negara, N., and Stroulia, E, : Automated Acceptance Testing of JavaScript Web Applications, Working Conference on Reverse Engineering(WCRE), P.318-322(2012).
20 Ruth, M., Sehun, Oh., Loup, A., Horton, B. et al.: Towards Automatic Regression Test Selection for Web Services,Annual International Computer Software and Applications Conference,Vol.2, P.729-736(2007).