情報処理学会論文誌 Vol.59 No.2 415–428 (Feb. 2018)

RFBキャプチャ/リプレイに基づく VDI対応監視制御システムの回帰テスト効率化方式

阿倍 博信[1,a)] 鶴崎 真理子[1] 塚原 研[1]

受付日 2017年5月9日, 採録日 2017年11月7日

概要：本論文では，VDI 対応監視制御システムの回帰テストにおいて，画面伝送プロトコルである RFB のキャプチャ/リプレイにより効率化を行う方式について述べる．方式設計にあたり，複数のスクリーンに対応したアプリケーションのスクリーン間の同期制御や通信トラフィックのモニタリングにより記録時と再生時の画面遷移のタイミング制御を行う方式を採用した．本方式を適用した VDI 対応監視制御システムの評価システムを開発し，自動再生時のロバスト性，キャプチャ画像の再現性について評価した結果，その有効性について確認することができた．

キーワード：回帰テスト，VDI，監視制御システム，RFB，キャプチャ/リプレイ，トラフィックモニタリング

An Efficient Method of Regression Testing of a VDI Compatible Supervisory Control System Based on the RFB Capture/Replay

Hironobu Abe[1,a)] Mariko Tsurusaki[1] Ken Tsukahara[1]

Received: May 9, 2017, Accepted: November 7, 2017

Abstract: In this paper, we describe an efficient method of regression testing of a VDI compatible supervisory control system using the RFB capture/replay, such as a screen transmission protocol. In the designing phase, we adopted a method of controling the timing of screen transition of capturing/replaying time by synchronization control between the screen of the application corresponding to multiple screens, and the monitoring of communication traffic. The results to develop a prototype system of a VDI compatible supervisory control system that implements proposed method, the evaluation experiment about robustness of the automatic playback and reproducibility of the captured images, we were able to confirm its effectiveness.

Keywords: regression testing, VDI, supervisory control system, RFB, capture/replay, traffic monitoring

1. はじめに

物理サーバの台数削減によるコスト削減要求や，東日本大震災以降の BCP（Business Continuity Planning）対策へのニーズの高まりから仮想マシン技術を利用したサーバ仮想化技術が注目されている [1]．現在，サーバ仮想化技術は普及期に入り大規模システムへの導入が可能になったことから，電力系統制御システムをはじめとする監視制御システムにおいても，実システムへの適用が始まっている [2], [3]．

監視制御システムは，システムの大規模化，ネットワーク化にともない，不正アクセスや情報漏えいなどの様々な脅威に直面しており，サイバー攻撃を受けた場合のサービス提供の影響は大規模かつ広範囲になることが予測されている [4]．たとえば，電力系統制御システムでは，2020 年 4 月に計画されている発送電分離に対して，託送機能を含めたシステムの大規模な機能拡張が要求されている [5]．

上記背景のもと，次世代の監視制御システムでは，デスクトップ仮想化技術である VDI（Virtual Desktop Infras-

[1] 三菱電機株式会社
Mitsubishi Electric Corporation, Kamakura, Kanagawa 247–8501, Japan
[a)] Abe.Hironobu@cs.MitsubishiElectric.co.jp

tructure）[1] の導入が検討されはじめている．VDI の利点として，HMI[*1]端末側にデータが残らないため，機密，個人情報の漏えい防止などのセキュリティ強化が期待できる．また，PC のハードウェアコストは大幅に低価格化した一方で，アプリケーション管理，セキュリティ強化，ユーザ対応など管理コストは確実に増えており，VDI の導入により運転員側の HMI 端末を仮想化することで，集中管理による管理の効率化が可能となる．

VDI 対応により，集中管理による端末環境の運用管理コストの削減を見込むことができるが，システムの品質管理部門では，1 カ月に 1 回の頻度でセキュリティ強化によるプラットフォーム更新のためのリリースにむけて，また，半年に 1 回の頻度でアプリケーションの機能追加のためのリリースにむけて，それぞれ回帰テストを実施する必要があり，それにともなう回帰テストの工数増大が課題となっている．

システムの品質管理部門は，主に実環境または模擬環境上でのシステムテストを担当しており，システムに容易にアドオン可能な回帰システム環境が求められている．そこで，今回，システムの品質管理部門での回帰テストへの適用を想定し，VDI で採用されている画面伝送プロトコルのキャプチャ/リプレイに基づき VDI 対応監視制御システムの回帰テストを効率化する方式を提案する．今回提案する方式では，VDI 対応システムでの回帰テストの効率化を実現するために，VDI で採用されている画面伝送プロトコルのキャプチャ/リプレイを利用することで，システムに回帰テスト環境を容易にアドオンすることが可能となる．今回は画面伝送プロトコルとして，広く VNC（Virtual Network Computing）で採用されている RFB（Remote Frame Buffer）[6] プロトコルを対象として，VNC クライアントに GUI 操作のキャプチャ機能およびリプレイ機能を追加する方針とした．

また，RFB プロトコルはベストエフォート通信が前提のため，キャプチャ時とリプレイ時でサーバやネットワークの状態が異なる場合は，マウス，キーボードなどの GUI 操作やエビデンス確認用の画面キャプチャと画面更新のタイミングが前後してしまう可能性がある．このような場合，テストシナリオの自動再生やエビデンス確認用の画面キャプチャが正常動作しない可能性が出てくる．

さらに本論文で対象とする監視制御システムの HMI 端末では，下記の理由からマルチスクリーン対応が前提となっている．

- 画面呼び出し操作や画面遷移時が 1 スクリーンにとどまらず，複数のスクリーンに影響を与えるため
- 1 つの業務を行うために 2 画面をペアで運用するケースがあるため
- スクリーンごとに運用用途が決められていることが多いため

そのため，スクリーン間でのタイミング同期制御も必要になってくる [7], [8], [9], [10]．

回帰テストの効率化において，テストシナリオの自動再生やエビデンス確認用の画面キャプチャが正常動作しないシナリオは，人手で作業を実施する必要があるため，自動再生時のロバスト性やキャプチャ画像の再現性は重要な要素の 1 つであると考えられる．

対応として，今回，複数スクリーン対応を前提にテストシナリオ自動再生のロバスト性やキャプチャ画像の再現性を向上させるために，リプレイ時には RFB プロトコルのフレームバッファ更新応答や HMI 端末の受信トラフィックのモニタリング結果に基づき，スクリーン間の協調制御によりタイミングを調整する方法を提案する．

本論文では，2 章で関連研究，3 章で要求条件，4 章で回帰テストの効率化方式，5 章で評価システムの開発，6 章で評価，7 章で考察について述べ，最後にまとめを行う．

2. 関連研究

本論文で対象とする VDI 対応監視制御システムの特徴の 1 つとして，オープンプラットフォーム対応があげられ，想定 OS として Linux，ウィンドウシステムとして X，開発言語として C を前提として検討を進めていく．また，前提条件として，画面伝送プロトコルとして RFB を採用し，システムの品質管理部門における回帰テストへの適用を想定して，数百～数千台規模のマルチスクリーン対応の HMI 端末から構成される VDI システムに容易にアドオン可能な回帰テスト環境の構築に関連する研究について述べる．

回帰テストとは，ソフトウェアの機能追加，改修などにより，ソフトウェアが改変された際に，変更が加わらない箇所についても繰り返しテストを行うことでソフトウェアの品質を確保するテストであり，繰返し回数が多いほど自動化の効果が期待できる．また，ソフトウェアに改変がなくても，セキュリティ強化によるプラットフォーム更新時などにも必要となるテストであり，自動化による効率化が必要とされている．

一般的に回帰テストの自動化対象としては，テストシナリオ自動再生，テスト結果自動確認の 2 つの工程があげられるが，本論文では前者のテストシナリオ自動再生を対象とする．

回帰テストにおけるテストシナリオの自動再生は，一般的に GUI 操作のキャプチャ/リプレイが使用される．以下，文献 [11] で紹介されている代表的なキャプチャ/リプレイツールについて説明する．

HP Unified Functional Testing [12]，Oracle Functional Testing [13], Rational Functional Tester [14], SilkTest [15], Visual Studio [16] はいずれも回帰テストのシナリオ自動再

[*1] Human Machine Interface：監視制御システムにおけるオペレータとのインタフェース

生機能を持つ商用の GUI 操作のキャプチャ/リプレイツールであり，マルチスクリーンには対応しているが，実環境にインストールして使用することを前提としているため，VDI システムに容易にアドオンして使用するという本論文の目的には適さない．

Selenium [17] は，Web システムを対象としたオープンソースのテストツールであるが，今回対象とする監視制御システムは，Web システムを前提としたものではないため，本論文の目的には適さない．

また，Abbot [18]，Jacareto [19]，Pounder [20] はいずれも Java/Swing を対象としたオープンソースの GUI 操作のキャプチャ/リプレイツールであるが，実環境へのインストールが前提であるのと，対象アプリケーションが Java/Swing に限定されるため，本論文の目的には適さない．

本論文では，VDI システムに容易にアドオン可能な回帰テスト方式の確立を目的としており，その観点から，VDI システムで採用されている画面伝送プロトコルである RFB をベースとするキャプチャ/リプレイ方式がポイントとなる．

rfbproxy [21], VNC Reflector [22] はプロキシとして VNC サーバと VNC クライアントの中間に入り，RFB プロトコルを中継する機能を持っており，一般的に複数の VNC クライアント間でセッションを共有するために使用される．また，共有したセッションを FBS（FrameBuffer Stream）[23] 形式で出力する機能を持っているが，この機能は共有したセッションのスクリーンを画面キャプチャして記録することが目的のため，本論文の目的には適さない．

VNCplay [24] は，VNC クライアントを機能拡張し，RFB のキャプチャ/リプレイを実現したツールである．VNCplay では本論文の目的と同様，セッションを記録し，リプレイすることで回帰テストでのテストシナリオの自動再生を実現している．

VDI システムに対する回帰テストの観点から先行研究である VNCplay が考慮できていない点として以下の 2 点があげられる．1 点目は，RFB プロトコルはベストエフォート通信が前提のため，キャプチャ時とリプレイ時でマウス，キーボードなどの GUI 操作やエビデンス確認用の画面キャプチャと画面更新のタイミングが前後してしまう可能性があり，このような場合，テストシナリオの自動再生やエビデンス確認用の画面キャプチャが正常動作しない可能性が出てくる点である．2 点目は，RFB プロトコルはシングルスクリーンでの動作が前提のため，監視制御システムの HMI 端末の前提条件である複数スクリーン間でのタイミング同期制御も必要になってくる点である．

今回，リプレイ時に RFB プロトコルのフレームバッファ更新応答や HMI 端末の受信トラフィックのモニタリング結果に基づき，スクリーン間の協調制御によりリプレイのタイミングを調整する方法を導入することで，複数スクリーン対応を前提としたテストシナリオ自動再生のロバスト性やキャプチャ画像の再現性の向上を図る．

3. 要求条件

前述したとおり，今回対象とする電力系統制御システムをはじめとする監視制御システムの前提条件として，マルチベンダ対応があげられ，ベースとする VDI システムは，特定のベンダに依存しないオープンプラットフォーム対応が前提となる．

今回，仮想化基盤としては，次世代監視制御システムにおいて，サーバ仮想化技術として標準的に採用された Xen [25]，または Linux カーネルに標準搭載されている KVM（Kernel-based Virtual Machine）[26] を，画面伝送プロトコルとしては，広く VNC で採用されている RFB を選択した．

以上をふまえて，監視制御システムのうち回帰テストの自動化の効果が見込める最大で数千台規模のクライアントへの対応が必要な大規模システムが前提となる電力系統制御システムを対象として，要求条件を検討する．

(1) システムの応答性
クライアントとサーバ間は必要十分な帯域を持った専用ネットワークで接続されていることを前提としているが，サーバにトラフィックが集中した場合などを想定し，通常時の標準応答時間に加えて最大 10 秒までレスポンスの低下を許容する．

(2) マルチスクリーン対応
文献 [7] は基本的な電力系統制御システムの監視制御室のイメージであるが，オペレータが操作する HMI 端末は最大 3 面のマルチスクリーン対応が基本構成となっている．今回も要求条件として，マルチスクリーン対応を前提とし，最大フル HD（$1,920 \times 1,080$）サイズのスクリーンを最大 3 面まで対応する必要がある．

4. 回帰テストの効率化方式

3 章で設定した要求条件をふまえて，回帰テストの効率化方式について検討した．

4.1 対象アプリケーションの分析

電力系統制御システムの HMI アプリケーションを対象として，回帰テストのテストシナリオの自動再生の観点から，アプリケーションの分析を行った．以下に分析結果について示す．

- 基本的に，スクリーン上に表示されているオブジェクトの選択またはボタンのクリックにより，処理が実行され，画面更新が発生する．
- 処理結果を表示するスクリーンは，マウス操作を行ったスクリーンとは限らず，全画面が更新される場合も多い．
- GUI 操作のキャプチャ/リプレイを行った際に，画面

更新のタイミングは前後する場合でも表示内容や表示位置は完全に再現することが可能である．

4.2 基本方針

これまでの議論をふまえ，今回提案する回帰テスト効率化方式の基本方針について整理する．

- システムの品質管理部門における回帰テストへの適用を想定し，最大で数千台規模のマルチスクリーン対応の HMI 端末から構成された VDI システムに容易にアドオン可能な回帰テスト環境の構築を目的とする．
- 文献 [27] で提案した RFB ベースのキャプチャ/リプレイ方式をベースに，初回実行時にマウス，キーボードなどの GUI 操作，および GUI 操作後の画面キャプチャ操作をシナリオとして記録しておき，2 回目以降は記録したシナリオを自動再生することにより，工数削減を目指す．
- サーバへのトラフィック集中時の負荷変動を想定し，RFB 通信の受信トラフィックをモニタリングすることで，シナリオ再生時の画面更新のタイミングの変動に対するロバスト性の向上，複数スクリーン間の画面遷移の同期制御を考慮した方式とする．
- システム要件として，システムの応答時間の目標値は通常時の標準応答時間に加えて最大 10 秒まで，HMI アプリケーションは最大フル HD（$1,920 \times 1,080$）サイズのスクリーンを最大 3 面までの対応，とする．

4.3 提案方式の概要

図 1 に本論文で提案する RFB ベースのキャプチャ/リプレイ方式の概要について示す．

図 1 を見れば分かるとおり，提案方式は主に VDI Client 上で動作するアプリケーションとして構築される．VDI Server 側には回帰テスト用のアプリケーションを特別にインストールする必要がなく，VDI Client としてテスト用の HMI 端末を追加するだけで回帰テスト環境の構築が完了するため，システムの品質管理部門における回帰テストに適した方式である．

以下，VDI Client 上で動作する 2 つのアプリケーションの機能概要について示す．

RFB Capture Replay（RCR） VDI Client 上で動作する VNC クライアントの拡張機能として，マウス操作，キーボード操作などの GUI 操作をシナリオとして記録しておき，自動再生時には，記録したシナリオに従い GUI 操作を再生することで，回帰テストの効率化を行う．また，シナリオ記録時に設定したタイミングで，シナリオ再生時にスクリーンを画面キャプチャしてイメージデータとして記録する機能を持つ．記録，再生は，RFB レベルで実行することで，VDI Server 上で動作するアプリケーションを直接制御できる．記録，再生の対象とする RFB 命令は，`KeyEvent`：キーボードイベント，`PointerEvent`：マウスイベント，`FramebufferUpdateRequest`：画面更新命令，とする．また，シナリオ記録時に RFB 通信のうち画面更新メッセージである `FramebufferUpdate` に着目し，受信したデータの受信トラフィックをログファイルに記録しておき，シナリオ再生時に参照する機能を持つ．

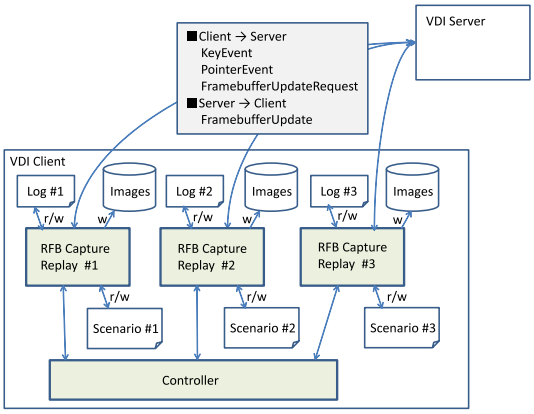


図 1 RFB キャプチャ/リプレイ方式の概要
Fig. 1 Overview of capture/replay method of RFB.

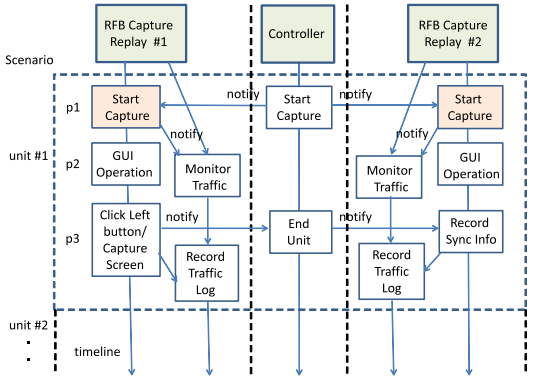


図 2 シナリオ記録の流れ
Fig. 2 Flow of scenario recording.

Controller RFB 通信の受信トラフィックを記録した RCR との連携により，シナリオ再生時は，記録時と再生時の画面遷移のタイミング制御や複数のスクリーンに対応したアプリケーションのスクリーン間を同期制御する機能を持つ．

4.4 シナリオ記録の流れ

シナリオ記録の処理フローは，VDI Client 上で，回帰テストの全体制御を行う `Controller` と対象アプリケーションの GUI 制御を行う `RFB Capture Replay`（`RCR`）をスクリーンごとに起動しておき，回帰テストのテスト手順に従う形で GUI 操作を行うことにより，その結果をシナリオとして記録する．**図 2** にシナリオ記録の流れについて示す．

図 2 に示したとおり，シナリオ記録は大きく 3 フェーズに分かれる．以下，フェーズごとの処理内容について説明

する．

4.4.1 フェーズ 1：シナリオ記録開始

VDI Client 上でユーザ操作により「シナリオ記録開始」が実行されると，シナリオ記録/自動再生の全体制御を行う Controller から，スクリーン単位での GUI 制御を行う RCR に対して，「シナリオ記録開始」が通知され，それぞれタイマを初期化してシナリオ記録を開始する．RCR は対象とするアプリケーションのスクリーン数だけ起動する．たとえば図 2 は 2 スクリーン構成のため，RCR を 2 つ起動する．並行して，RFB 通信の FramebufferUpdate メッセージの受信トラフィックのモニタリングを開始する．

4.4.2 フェーズ 2：GUI 操作記録

VDI Client 上でスクリーン単位で GUI 制御を行う RCR で，あらかじめ設定された回帰テストのテスト手順に従い GUI 操作を行う．操作結果は RFB 経由で VDI Server に通知され，処理結果も RFB 経由で RCR 上に反映される．GUI 操作の結果はフェーズ 1 で初期化したタイマのタイムスタンプとあわせてスクリーンごとにシナリオファイルに記録される．並行して，フェーズ 1 で開始した受信トラフィックのモニタリングを継続実施する．

4.4.3 フェーズ 3：ユニット記録終了

本論文において，シナリオは複数のユニットから構成されることを前提としており，ユニット単位でのスクリーン間の同期を目指す．ユニット切れ目の検出は画面更新の可能性の高い特定イベントの発生，テスト結果のエビデンス確認など回帰テストにおけるテスト実行の区切りとなる画面キャプチャの GUI による指定を用いる．

VDI Client 上で，RCR で事前定義しておいた特定のマウスイベント（「左ボタン」クリック）が発生した場合や，ユーザ操作により「画面キャプチャ」が実行された場合は，ユニット記録が終了となり，シナリオファイルにタイムスタンプにあわせてその内容を記録するとともに，継続して次のユニットのシナリオ記録を自動的に開始する．

「画面キャプチャ」が指定された場合は，そのタイミングでスクリーンを画面キャプチャしてイメージデータとして記録する処理を実行する．

同時に，「特定イベント」または「画面キャプチャ」の発生によりユニットが終了となるため，対象ユニットの受信トラフィックの集計結果をログファイルに記録する．また，ユニット終了の通知は，Controller を経由して他の RCR にもその内容が通知され，その時点のタイムスタンプにあわせて同期情報としてその内容をシナリオファイルに記録するとともに，対象ユニットの受信トラフィックの集計結果をログファイルに記録する．

一方，VDI Client 上で，ユニット記録の途中でも，ユーザ操作により「シナリオ記録終了」が実行されるとその時点でシナリオ記録が終了となる．

図 3 に RCR が出力するシナリオファイルの例，**図 4** にログファイルの例について示す．

```
0,p,0,75,274,,
187,p,0,75,273,,
47,p,0,76,273,,
171,p,1,76,273,,
47,p,0,76,273,,
234,f,1,0,0,1366,768
125,f,1,0,0,1366,768
1290,c,,,,,
359,k,1,20481,65513,,
15,k,1,20481,65507,,
31,k,1,20481,65505,,
203,f,1,0,0,1366,768
265,k,0,20481,80,,
63,k,0,20481,65513,,
15,k,0,20481,65505,,
16,k,0,20481,65507,,
1890,s,,,,,
```

図 3 シナリオファイルの例
Fig. 3 Example of scenario file.

```
1455,0,17543324
7785,0,8145677
24142,1,117738
14577,0,14617001
32458,0,24876113
```

図 4 ログファイルの例
Fig. 4 Example of log file.

図 3 において，シナリオファイルは 1 行で 1 イベントを示し，各行において，1 番目の項目は直前のイベントからの経過時間（msec），2 番目の項目はイベントの種別で p は PointerEvent，k は KeyEvent，f は FramebufferUpdateRequest を示し，3 番目の項目以降は，各イベントの引数を示す．各イベントの引数は RFB の仕様に準拠した固定長のデータとなり，p で 5 バイト，k で 7 バイト，f で 9 バイトの値となる．

また，2 番目の項目に c が入っていた場合は「画面キャプチャ」を，s が入っていた場合は同期情報をそれぞれ示す．

さらに，特定イベントの検出はマウスの左クリックの場合，2 番目の項目であるイベントの種別が p で，3 番目の項目に 1（down）が入り，その次のイベントで X 座標，Y 座標がほぼ移動せず 0（up）が入った場合となる．

図 4 において，ログファイルは 1 行で 1 ユニットを示し，各行において，1 番目の項目はユニット長（msec），2 番目の項目はユニットの種別で 0 は特定イベントの発生によるユニットの終了，1 は画面キャプチャによるユニットの終了，3 番目の項目は受信トラフィックの集計結果（bytes）をそれぞれ示す．

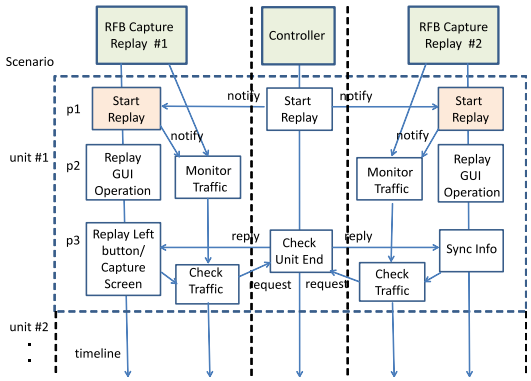


図 5 シナリオ自動再生の流れ
Fig. 5 Flow of automatic replay of scenario.

図 3，図 4 を見ても分かるとおり，シナリオファイル，ログファイルとも CSV 形式を前提としているため，表計算ソフトなどで読み込んで内容を確認することが可能である．

4.5 シナリオ自動再生の流れ

シナリオ自動再生の処理フローは，シナリオ記録と同様，VDI Client 上で，回帰テストの全体制御を行う Controller と対象アプリケーションの GUI 制御を行う RCR をスクリーンごとに起動しておき，シナリオ記録時に記録したシナリオを読み込み，シナリオ記録時の GUI 操作を自動再生する．**図 5** にシナリオ自動再生の流れについて示す．

図 5 に示したとおり，シナリオ自動再生は大きく 3 フェーズに分かれる．以下，フェーズごとに，処理内容の詳細について説明する．

4.5.1 フェーズ 1：シナリオ再生開始

まず，事前準備として，スクリーン単位での GUI 制御を行う RCR において，シナリオ記録時に記録しておいたシナリオファイルとログファイルを読み込んでおく．

次に，VDI Client 上でユーザ操作により「シナリオ再生開始」が実行されると，Controller から RCR に対して，「シナリオ再生開始」が通知され，それぞれタイマを初期化してシナリオ再生を開始する．RCR は対象とするアプリケーションのスクリーン数だけ起動する．たとえば図 5 は 2 スクリーン構成のため，RCR を 2 つ起動する．並行して，RFB 通信の FramebufferUpdate メッセージの受信トラフィックのモニタリングを開始する．

4.5.2 フェーズ 2：GUI 操作再生

VDI Client 上でスクリーン単位で GUI 制御を行う RCR で，読み込んだシナリオファイルを解析し，タイマのタイムスタンプのタイミングにあわせて，RFB 要求を実行し，RFB 経由で VDI Server に通知する．また，VDI Sever での処理結果も RFB 経由で RCR 上に反映される．並行して，フェーズ 1 で開始した受信トラフィックのモニタリングを継続実施する．

4.5.3 フェーズ 3：ユニット再生終了

VDI Client 上で，RCR にてタイマのタイムスタンプのタイミングが読み込んだシナリオファイルの「特定イベント」や「画面キャプチャ」のタイムスタンプのタイミングに到達した場合，タイマ再生を一時停止し，読み込んだログファイルの内容と受信トラフィックのモニタリング結果に基づきタイミング同期制御を行い，「特定イベント」または「画面キャプチャ」に到達したことを「再生継続要求」として Controller に通知する．

RCR における受信トラフィックモニタリングによるタイミング同期制御の流れは，ログファイルから読み込んだ対象ユニットの受信トラフィックの集計結果と，モニタリング中の受信トラフィックを比較し，受信トラフィックの数値が，ログファイルから読み込んだ数値に設定した閾値を乗算し，100 で除算した数値に到達したタイミングで，「再生継続要求」を Controller に通知する．閾値は 0〜100 の数値データとして設定する．

他の RCR でも，ほぼ同じタイミングでタイマのタイムスタンプが同じイベントの同期情報に到達するため，タイマ再生を一時停止し，読み込んだログファイルの内容と受信トラフィックのモニタリング結果に基づくタイミング同期制御を行い，その結果を「再生継続要求」として Controller に通知する．

Controller では，制御対象の全 RCR から受信した「特定イベント」または「画面キャプチャ」の到達による「再生継続要求」を受信した時点で，対象ユニットの再生が終了したと判断し，制御対象の全 RCR に対して「再生継続応答」を戻す．

RCR では，「再生継続応答」を受信すると，タイマ再生を再開し，シナリオの内容に従って特定イベント処理または画面キャプチャ処理を行う．この処理をもってユニット再生が終了となり，シナリオファイルの内容に従って次のユニットの再生を自動的に継続する．

一方，VDI Client 上で，ユニット再生の途中でも，ユーザ操作により「シナリオ再生終了」が実行されるとその時点でシナリオ再生が終了となる．

5. 評価システムの開発

これまでの検討結果に基づき，3 章で設定した要求条件を考慮した形で，VDI システムに対応した監視制御システムにアドオンが可能な回帰テスト環境の評価システムを開発した．

5.1 システム構成

図 6 に回帰テスト環境を適用した VDI システムに対応した監視制御システムの評価システム構成について示す．

評価システムでは，回帰テストにおけるシナリオ自動再生を目的として，マルチスクリーンに対応した VDI Client 上

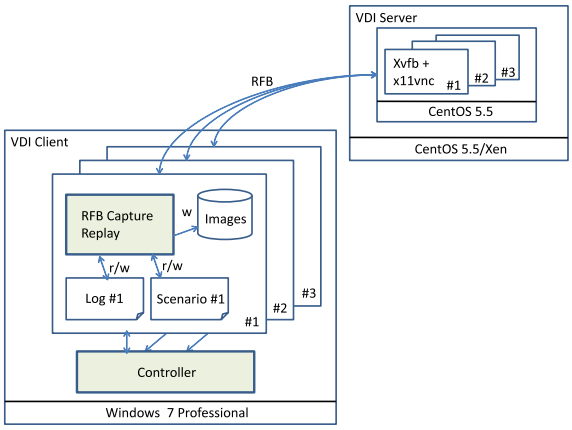


図 6 評価システム構成
Fig. 6 Prototype system configuration.

表 1 VDI Client の仕様
Table 1 Specification of VDI Client.

プロセッサ	intel Xeon E3-1231 v3
システムクロック	3.4 GHz quad-core
RAM	8 GB
OS	Windows 7 Professional

表 2 VDI Server の仮想マシンの仕様
Table 2 Specification of virtual machine of VDI Server.

プロセッサ	intel Xeon E5530
システムクロック	2.4 GHz 2 core
RAM	2 GB
OS	CentOS 5.5

に RCR と Controller の機能を実装した専用の VDI Client を準備した．**表 1** に VDI Client の仕様について示す．

また，回帰テストの対象となる監視制御アプリケーションは，Domain-0 として CentOS 5.5 をインストールした PC サーバ上に，仮想化環境として Xen 環境を導入して仮想マシンを構築し，仮想マシン上のゲスト OS として CentOS 5.5 をインストールし，Xvfb（X virtual framebuffer）[28] を導入して X Window System の仮想ディスプレイ環境を構築するとともに，監視制御アプリケーションを準備した．さらにゲスト OS の 3 面のスクリーンそれぞれに VNC Server として x11vnc [29] を割り当てた．**表 2** に VDI Server の仮想マシンの仕様について示す．

また，ネットワーク構成としては，VDI Client，VDI Server を 1000BASE-T ネットワークで相互接続する構成とし，VDI Client と VDI Server 間の RFB 接続におけるスクリーン圧縮方式は「Hextile」を選択した．

5.2 RCR（RFB Capture Replay）

RCR を，オープンソースの VNC ソフトウェアである TightVNC [30] Viewer の拡張機能として，VDI Client 上に実装した．

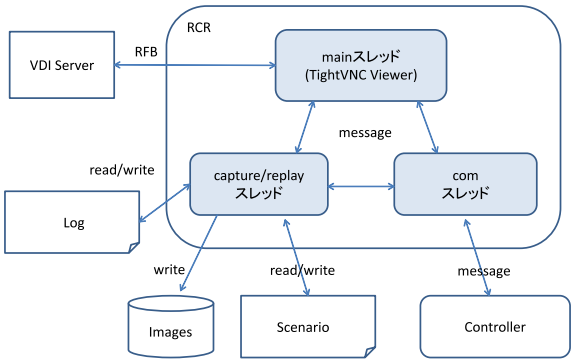


図 7 RCR のモジュール構成
Fig. 7 Module configuration of RCR.

RCR の機能について下記のとおり説明する．

- VDI Server と RFB 経由で接続し，GUI 操作をシナリオファイルとして記録する．
- 記録したシナリオファイルを読み込み，その内容に従いシナリオを自動再生する．
- シナリオ記録時は RFB 通信の FramebufferUpdate メッセージの受信トラフィックをモニタリングしログファイルに出力する．
- シナリオ再生時はタイミング同期制御のために，RFB 通信の FramebufferUpdate メッセージの受信トラフィックをモニタリングする．

RCR のモジュール構成について，TightVNC Viewer に対する拡張機能を中心に**図 7** に示す．

図 7 に示したとおり，RCR の内部は「main」，「capture/replay」，「com」の 3 スレッドで構成される．以下，RCR の処理フローについて示す．

(1) 初期化処理

main スレッド TightVNC Viewer の main スレッド．VDI Server 上の VNC Server と接続するとともに，capture/replay スレッド，com スレッドの起動を行う．また，初期化処理として capture/replay スレッドで取り扱うシナリオファイルをオープンしておく．

(2) シナリオ記録処理

com スレッド Controller から「シナリオ記録開始」メッセージを受信し，capture/replay スレッドに伝える．

capture/replay スレッド タイマを初期化する．

main スレッド VDI Server との RFB 通信をポーリングし，シナリオファイルやログファイルに記録すべき RFB 命令を見つけた場合は，capture/replay スレッドに通知する．

capture/replay スレッド main スレッドから受けた RFB 命令が KeyEvent：キーボードイベント，PointerEvent：マウスイベント，FramebufferUpdateRequest：画面更新命令の場合

は，その内容を内部タイマのタイムスタンプ情報とあわせてシナリオファイルに記録する．また，RFB 命令が FramebufferUpdate：画面更新メッセージの場合は，受信したデータの受信トラフィックを内部バッファに記録する．また，受信した RFB 命令が，特定イベントと一致した場合や，main スレッド経由で「画面キャプチャ」を受けた場合，シナリオファイルにその内容を記録するとともに，内部タイマのタイムスタンプ情報と内部バッファの受信トラフィックの情報をログファイルに出力するとともに，その内容を com スレッドに通知する．「画面キャプチャ」の場合は，main スレッドの保持するスクリーンのスナップショット画像を可逆圧縮の画像フォーマットである bmp 形式で記録する．

com スレッド　capture/replay スレッドから通知された内容を Controller に通知する．また，Controller から同期情報を受信するとその内容を capture/replay スレッドに通知する．

capture/replay スレッド　受信した同期情報を内部タイマのタイムスタンプ情報とあわせてシナリオファイルに記録する．その際にも，内部タイマのタイムスタンプ情報と内部バッファの受信トラフィックの情報をログファイルに出力する．

(3) シナリオ自動再生処理

com スレッド　Controller から「シナリオ再生開始」メッセージを受信し，capture/replay スレッドに伝える．

capture/replay スレッド　タイマを初期化する．シナリオファイルから読み込んだ RFB 命令を内部タイマのタイムスタンプにあわせて main スレッドに通知する．また，ログファイルから読み込んだ受信トラフィックの情報を読み込む．

main スレッド　capture/replay スレッドから受けた RFB 命令を VDI Server に繰り返し通知するとともに，VDI Server との RFB 通信をポーリングし，シナリオファイルやログファイルに記録すべき RFB 命令を見つけた場合は，capture/replay スレッドに通知する．

capture/replay スレッド　main スレッドから受けた RFB 命令が FramebufferUpdate:画面更新メッセージの場合は，受信したデータの受信トラフィックを内部バッファに記録する．また，受信した RFB 命令が，特定イベントと一致した場合や，main スレッド経由で「画面キャプチャ」を受けた場合，または同期情報の場合，タイマ更新を一時停止するとともに，ログファイルから読み込んだ対象ユニットの受信トラフィックの集計結果と，モニタリング中の受信トラフィックの比較処理によるタイミング同期制御が終了した時点で，その内容を com スレッドに通知する．

com スレッド　capture/replay スレッドから通知された内容を Controller に通知する．また，Controller から「シナリオ再生再開」通知を受信し，その旨を capture/replay スレッドに通知する．

capture/replay スレッド　「シナリオ再生再開」通知を受けると，タイマ更新の一時停止を解除するとともに，特定イベントの場合は，main スレッド経由で VDI Server に RFB 命令を送信し，「画面キャプチャ」の場合は，main スレッドの保持するスクリーンのスナップショット画像を可逆圧縮の画像フォーマットである bmp 形式で記録する．特定イベントの同期情報の場合は，一時停止解除のみを行う．画面キャプチャの同期情報の場合，実際に画像ファイルを出力するかどうかは設定により処理を変更する．また，受信トラフィックを記録する内部バッファをクリアする．

5.3 Controller

4 章の設計結果をもとに，VDI Client で動作し，同じ VDI Client 上で動作する複数の RCR を連携制御する機能を持つ Controller を Windows アプリケーションとして開発した．

6. 評価

6.1 評価指針

開発した評価システムを用いて，評価用シナリオを準備し，3 章で設定した要求条件および 4 章で提案した方式に基づき，下記の項目についての評価を行った．

(1) **回帰テストの自動再生時のロバスト性**

提案方式は，サーバへのトラフィック集中時の負荷変動を想定し，通常時の標準応答時間に加えて最大 10 秒までレスポンスの低下を許容する形で，フル HD サイズのスクリーン 3 面構成の HMI 端末でのシナリオ自動再生のロバスト性確保を目標としている．評価シナリオを用いて，シナリオの完走回数や再生時間などのロバスト性の評価を行う．

(2) **自動再生時のキャプチャ画像の再現性**

提案方式における受信トラフィックのモニタリング結果に基づく，フル HD サイズのスクリーン 3 面構成の HMI 端末におけるスクリーン間の協調制御の評価を目的として，評価シナリオを用いて，画面キャプチャを行い，画像処理に基づくキャプチャ画像の再現性の評価を行う．

6.2 評価用シナリオの準備

評価にあたり，電力系統制御システムの回帰テストへの適用を想定して，電力系統制御システムにおける通常時の

操作を想定し，あらかじめ決められた操作手順に従って，4.1 節で実施した対象アプリケーションの分析結果をすべてカバーすることを前提として，系統制御アプリケーションの GUI を操作する手順を評価用シナリオとして設定する．

本論文で対象とする監視制御システムの機能は，全システムで共通となる基本機能とユーザ要求によりシステムごとに開発する拡張機能から構成される．また，回帰テストの対象としては基本機能と拡張機能を加えて，小規模システムで 50 シナリオ，大規模システムで 1,000 シナリオ程度の確認が必要と考える．評価用シナリオの選定にあたり，対象とする監視制御システムの HMI アプリケーションの機能を分析した結果，規模に関係なく全システムで共通となる基本機能として，スケルトン画面，系統状態監視画面，系統操作画面，状変記録画面から構成される 20 シナリオを抽出した．

4.1 節で分析したとおり，対象アプリケーションは，規模に関係なく同一の設計指針に基づき設計されていることから，抽出した基本機能のシナリオの中から，典型的かつ最も使用頻度の高いシナリオを抽出した結果，**表 3** に示す 7 シナリオを評価用シナリオとして選定した．拡張機能は，基本機能と同一の設計指針である，使用頻度が基本機能ほど高くない，の理由から今回は対象外とした．

評価用シナリオの作成時には，VDI Server は通常時での操作を想定し，評価用アプリケーション以外は動作させない状態で，シナリオの記録操作を行った．その際に，ユニットの終了時には必ず画面キャプチャを行う設定で操作の記録を行った．

表 3 評価用シナリオの概要
Table 3 Overview of the scenarios for evaluation.

シナリオ番号	GUI 操作			実行結果		
	左画面	中画面	右画面	左画面	中画面	右画面
1	-	a	-	A	A	A
	a	-	-	B	B	B
	a	-	-	A	A	A
2	-	a	-	A	A	A
	-	-	b	-	-	A
3	-	a	-	A	A	A
	a	-	-	A	-	A
	b	-	-	A	-	A
	b	-	-	A	-	-
	b	-	-	A	-	-
	b	-	-	A	-	A
4	-	a	-	A	A	A
	-	a	-	-	A	-
	-	a	-	-	A	-
	-	a	-	-	A	-
5	a	-	-	A	A	A
	-	a	-	A	A	A
6	b	-	-	A	A	A
	b	-	-	A	-	-
	b	-	-	-	A	-
	-	a	-	A	A	A
7	-	-	b	A	A	A
	-	-	c	-	-	A
	-	-	b	-	-	A

a：オブジェクト選択，b：ボタンクリック，c：スクロールバードラッグ，A：処理結果表示，B：点滅停止

6.3 予備実験

テストシナリオ自動再生のロバスト性やキャプチャ画像の再現性評価に向け，RCR の受信トラフィックモニタ機能で使用する閾値の適正値を求めるための予備実験を実施した．

具体的には，画面 A と画面 B の 2 画面から構成され，画面 A に表示されているボタンをクリックすると，設定された待ち時間経過した後で画面 B に自動的に遷移する評価用アプリケーションを準備した．次に，評価システム上にて，1 スクリーン構成で評価用アプリケーションを動作させ，シナリオ記録時は画面遷移の待ち時間を 5 秒に設定して操作記録を行った．再生時は，システム負荷の変動を想定して，画面遷移の待ち時間をシステムの標準応答時間の 2 倍の 20 秒に設定し，RCR の受信トラフィックモニタ機能で使用する閾値を（10%，30%，50%，70%，90%）につど変更する形で 5 回ずつシナリオの自動再生処理を試行し，シナリオの完走回数を計測した．

この際，下記の 2 つの条件を同時に満たすことを「シナリオ完走」と定義した．

(1) シナリオ記録時に記録した画面が記録順にすべて表示される．

(2) シナリオ記録時に記録した操作が記録順にすべて実行される．

表 4 予備実験の評価結果
Table 4 Evaluation results of preliminary experiments.

閾値（%）	完走回数（回）
10	0
30	0
50	3
70	5
90	2

シナリオの自動再生は画面再生と操作の連携が必須であり，画面表示が完了する前に操作を実行した場合や操作の前提となる画面表示が完了しない場合が発生するとシナリオは完走しない．

その結果を**表 4** に示す．

表 4 において，RCR の受信トラフィックモニタ機能で使用する閾値は，70%が最もシナリオの完走率が高かった．

6.4 シナリオの自動再生時のロバスト性の評価

シナリオ記録時と自動再生時でシステムの負荷が変動し

表 5 評価結果 (1)：シナリオの完走回数
Table 5 Evaluation results (1): Number of finished scenarios.

シナリオ番号	完走回数（回）			
	同期制御あり		同期制御なし	
	高負荷	通常負荷	高負荷	通常負荷
1	5	5	5	5
2	5	5	5	5
3	5	5	5	4
4	5	5	5	5
5	5	5	5	5
6	5	5	0	5
7	4	4	5	4

た場合のシナリオ自動再生機能のロバスト性について評価した．

具体的には，6.2 節で準備した評価用シナリオに対して，VDI Server のシステム負荷を 2 段階（通常負荷：シナリオ記録時と同じ，高負荷：シナリオ再生中にファイル I/O の負荷と CPU 負荷を上げる）に条件を設定した．次に，各負荷に対して，RCR に提案方式である受信トラフィックモニタによるタイミング同期制御を行った場合と行わなかった場合で各条件に対して 5 回ずつシナリオの自動再生処理を試行し，シナリオの完走回数を計測した．また，シナリオが完走できた場合は，タイミング同期制御を行った場合について，通常負荷と高負荷に対して，応答時間を計測するために，記録時と再生時のユニットごとの再生時間を計測し，その差をユニットごとの再生延長時間として算出した．同様に，通常負荷と高負荷に対して，その再生時間を計測し，記録時との比を再生時間延長率として算出した．また，RCR の受信トラフィックモニタ機能で使用した閾値は，予備実験の結果に基づき，70%とした．

VDI Server のシステム負荷の調整はオープンソースの負荷発生ツールである stress [31] を使用し，高負荷状態では，`-d --hdd-bytes 128M` オプションを用いて 128 MB の書き込みを行うプロセスを 1 つ実行するとともに，`-c 1` オプションを用いて CPU 負荷を追加する形で実施した．

また，2 章で説明したとおり，提案方式はシステムの品質管理部門における回帰テストへの適用を想定し，数百〜数千台規模のマルチスクリーン対応の HMI 端末から構成される VDI システムに容易にアドオン可能な回帰テスト環境の実現を目的としている．そのため，サーバへのトラフィック集中時などに発生する VDI Server 側のディスク負荷，CPU 負荷の高負荷状態を模擬的に実現することにより，より実環境に近い環境での評価を行う．

その結果をそれぞれ**表 5**，**表 6**，**表 7** に示す．

表 5 において，タイミング同期制御を導入することにより，導入しなかった場合と比較して，シナリオ完走率が向上することが分かった．高負荷時のシナリオ 6 はタイミング同期制御がなしの場合は完走率が 0 であったが，提案方式の導入により，シナリオの完走率が改善された．

表 6 評価結果 (2)：ユニットごとのシナリオの再生延長時間
Table 6 Evaluation results (2): Extension time of playing period of scenarios per unit.

シナリオ番号	平均再生延長時間（秒）	
	高負荷	通常負荷
1	1.18	1.36
2	6.23	1.23
3	6.88	7.60
4	10.17	5.52
5	4.25	2.55
6	10.33	8.37
7	6.54	0.77

表 7 評価結果 (3)：シナリオの再生時間延長率
Table 7 Evaluation results (3): Extension rate of playing period of scenarios.

シナリオ番号	平均再生時間延長率	
	高負荷	通常負荷
1	1.27	1.29
2	3.48	1.40
3	3.67	3.84
4	5.90	3.79
5	2.93	2.02
6	5.13	4.26
7	24.71	3.86

表 6 において，ユニットごとの再生延長時間は，シナリオ 4 とシナリオ 6 の高負荷の場合を除き，平均値としては応答時間の目標値である標準応答時間+10 秒を満たしていることが分かった．未達成のシナリオを分析してみたところ，シナリオ 4 の高負荷の場合，全 12 ユニット中 5 個のユニットが，シナリオ 6 の高負荷の場合，全 12 ユニット中，4 個のユニットが，目標値の 10 秒を上回っており，最大値はシナリオ 4 で 28.4 秒，シナリオ 6 で 24.4 秒という結果となった．

表 7 において，シナリオ再生時間延長率は，シナリオに大きく依存することが分かった．また，負荷が高くなると平均再生時間も長くなる場合が多く，極端なケースとしては，シナリオ 7 では，高負荷時は平均再生時間が記録時の 24.71 倍かかるケースがあることが分かった．

6.5 自動再生時のキャプチャ画像の再現性の評価

評価システムを用いて，自動再生時のキャプチャ画像の再現性について評価した．前節の評価で完走したシナリオで，シナリオ記録時とシナリオ再生時で，同じタイミングでキャプチャしたキャプチャ画像を画素単位で比較することにより，その再現性について評価した．

具体的には，キャプチャ画像の画素一致率 $ConcordanceRate$ は，画素比較を行った結果合致した画素数 $MatchedPixels$ と画像全体の画素数 $AllPixels$

表 8 評価結果 (3)：キャプチャ画像の再現性
Table 8 Evaluation results (3): Reproducibility of captured images.

シナリオ番号	画素一致率（$ConcordanceRate$）			
	同期制御あり		同期制御なし	
	高負荷	通常負荷	高負荷	通常負荷
1	99.83	99.74	99.11	99.32
2	99.62	99.66	96.71	99.24
3	99.59	98.23	99.31	93.40
4	99.98	99.98	99.96	99.95
5	99.96	99.98	76.27	88.75
6	99.97	99.83	–	78.53
7	95.58	96.32	88.62	81.62

を用いて以下の数式を用いて算出した．

$$ConcordanceRate = \frac{MatchedPixels}{AllPixels} \times 100 \qquad (1)$$

また，$MatchedPixels$ の算出方式であるが，現実的には画面データの伝送時におけるエンコード方式としては高効率化が可能な不可逆変換が選択されることを想定し，各画素の RGB 値を人間の視覚に近い L*a*b*値[*2]にいったん変換し，以下の数式を用いて色差 ΔE を算出し，文献 [32] を参考に ΔE が印象レベルでは同じ色として扱える範囲（B 級許容差）である 6.5 以下の場合一致と判断する方式を用いる方針とした．

$$\Delta E = \sqrt{(\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2} \qquad (2)$$

キャプチャしたすべてのキャプチャ画像に対して，$ConcordanceRate$ を算出した結果，正常に記録できた場合は $ConcordanceRate$ が 90 以上，タイミングのずれなどにより正常に記録できなかった場合は $ConcordanceRate$ が 90 以下の値をとることが判明した．正常に記録できた場合でも $ConcordanceRate$ が 100 にならなかったのは，対象アプリケーションが時刻表示機能を持っており，正常に記録できても時刻表示部分に差異が発生したためである．

そこで，全体の試行回数に対して正常に記録できた回数の割合を算出し，その結果を**表 8** に示す．

表 8 において，提案方式であるタイミング同期制御を導入することにより，すべてのシナリオにおいて，導入しなかった場合と比較して負荷に関係なくキャプチャ画像の画素一致率の改善を確認することができた．

7. 考察

7.1 予備実験に対する考察

予備実験の結果，`RCR` の受信トラフィックモニタ機能で使用する閾値は，70%が最もシナリオの完走率が高かった．シナリオ自動再生を正常に動作させるためには，画面表示と操作の連携が必須であるが，他の閾値で両者の連携がうまくいかず，シナリオが完走できなかった理由として以下が考えられる．

(1) **閾値が低すぎた場合**
シナリオ再生時において，画面表示が完了しないうちに操作が行われてしまった可能性がある．

(2) **閾値が高すぎた場合**
画面データの伝送時におけるエンコード方式は高効率な不可逆変換を選択することが現実的であり，シナリオ記録時とシナリオ再生時で同じシナリオを再生しても，トラフィック量が異なる．そのため，閾値を高く設定すると，シナリオ再生時において，画面表示が完了してもトラフィック量が閾値に到達せず，操作が待ち状態のままで停止してしまった可能性がある．

7.2 シナリオの自動再生時のロバスト性の評価に対する考察

3 章で設定した要件および 4.2 節で設定した基本方針をふまえて，6.4 節の評価結果について考察する．

まず，シナリオの完走率であるが，VDI Server のシステム負荷の変動により，システムの応答時間が変動した場合でも，提案方式の特長である画面更新メッセージである `FramebufferUpdate` の受信トラフィックをスクリーンごとでモニタリングすることで，スクリーン間の同期制御ができており，ほぼ 100%のシナリオの完走率を得られることが確認できた．

課題としては，提案方式である受信トラフィックのモニタリングに基づくタイミング同期制御機能を導入しても，シナリオの内容や，再生時のシステム負荷，受信トラフィックモニタ機能で設定した閾値などに起因するタイミングの問題から，シナリオの自動再生に失敗する場合が残っている点である．

一般的に，自動再生が失敗したシナリオは，手動でテストを行う必要があるが，自動再生が失敗してしまうと，それ以降のシナリオの再生は無効となるため，今後は，キャプチャスクリーンの比較などによるテスト結果自動確認と組み合わせることで，シナリオの自動再生結果の確認を自動化することで，テスト作業の効率化が期待できる．

次に，シナリオの再生時間については，シナリオの内容に大きく依存することが分かった．評価実験の結果，シナリオの再生時間は，通常負荷の場合で記録時の 1.29〜4.26 倍，高負荷の場合で記録時の 1.27〜24.71 倍かかることが分かった．

特に，シナリオ 7 で高負荷の場合だと，通常負荷と比較して大幅に再生時間が延長してしまうケースがあることが分かった．表 3 を確認すると，シナリオ 7 は GUI 操作としてスクロールバーのドラッグ操作が入っているため，高負荷になりシステムの応答時間が長くなったのと，タイミング同期制御のためのイベントが連続的に発生したのが複

[*2] 明度を L*，色相と彩度を示す色度を a*，b*で表現する．

合したことが原因と考えられる．

また，ユニットごとの再生延長時間は，シナリオ単位での平均値でみるとほぼ応答時間の目標値である標準応答時間+10 秒を満たしているが，個々のユニット単位で見てみると，0 秒から 28.4 秒まで大きくばらついていることが分かった．ユニット単位では，最大 28.4 秒は目標値を大きく超えているが，提案方式はシナリオ単位で処理を行うため，実運用上は問題ないと判断した．

課題としては，シナリオの再生時間は，評価実験の結果，シナリオの内容や，再生時のシステム負荷，受信トラフィックモニタ機能で設定した閾値などにより，完走した場合でも，大きく変動してしまうことが分かり，現状は作業時間が見積もれない点である．

一般的には，回帰テストのシナリオ自動再生は，夜間にバッチ処理で行う場合が多く，再生時間の変動はあまり大きな問題にはならないと考えるが，再生時間の見積りのための評価用のシナリオを準備して再度評価実験を行い再生時間のモデル化を行うことで，再生時間の見積りが可能となる．

7.3 自動再生時のキャプチャ画像の再現性の評価に対する考察

3 章で設定した要件および 4.2 節で設定した基本方針をふまえて，6.5 節の評価結果について考察した結果，キャプチャ画像を正常に記録できた割合は，提案方式であるタイミング同期制御を導入することにより，すべてのシナリオにおいて，導入しなかった場合と比較して負荷に関係なくキャプチャ画像の画素一致率の改善を確認することができた．

ここでは，表 8 の結果をシナリオごとに考察する．まず，トラフィック同期制御なしで，シナリオ自動再生に失敗したシナリオを評価すると，シナリオ 3（通常負荷），シナリオ 6（高負荷），シナリオ 7（通常負荷）は，画素一致率がそれぞれ，93.40，0，81.62 から 98.23，99.97，96.32 に改善できていることが確認できた．

また，トラフィック同期制御なしで，シナリオ自動再生に成功した場合でも，シナリオ 5，6，7 は画素一致率が 90 以下の結果となり，再生タイミングのずれになどより，正常に記録できていなかったと判断できるが，提案方式によるトラフィック同期制御機能を導入することにより，画素一致率がすべて 90 以上の値に改善できていることが確認できた．

以上より，提案方式によるトラフィック同期制御機能を導入することにより，回帰テストの効率化の主要な課題の 1 つであるテスト結果の確認作業に対して，キャプチャ画像の比較によるテスト結果の自動確認による作業の効率化が期待できる．

7.4 提案方式の一般性に対する考察

7.1 節から 7.3 節までの考察をふまえて，提案方式の一般性について考察する．

予備実験の結果，提案方式において同期制御のポイントとなる RCR の受信トラフィックモニタ機能で使用する閾値は，低すぎる場合でも高すぎる場合でもシナリオの完走率が低下してしまうという課題があり，中間の 70%が適切であるという結果が得られた．

次に，対象とする監視制御システムの HMI アプリケーションの機能のうち，全システムで共通となる基本機能をから最も使用頻度の高いシナリオを評価用シナリオとして選定し，回帰テストの自動再生時のロバスト性，自動再生時のキャプチャ画像の再現性についてそれぞれ評価を行った．

4.1 節で分析結果を報告したが，監視制御システムの HMI アプリケーションは基本機能，拡張機能とも，基本的には選定した評価用シナリオと同じ設計指針のもとで設計されているため，提案方式を他のシナリオに適用する場合の一般性は高いと考える．

8. おわりに

本論文では，VDI 対応監視制御システムにおいて課題となっていた回帰テストの工数増大に対して，VDI で採用されている画面伝送プロトコルである RFB プロトコルのキャプチャ/リプレイにより，テストシナリオの自動再生を実現することで，効率化する方式を提案した．

方式設計にあたり，VDI 環境における回帰テスト方式の課題であるタイミングの変化に起因するテストシナリオの自動再生時のロバスト性の低下や監視制御システムの要件であるマルチスクリーンのタイミングの同期制御を考慮し，記録時に記録したデータに基づき，リプレイ時に RFB プロトコルのフレームバッファ更新応答や HMI 端末の受信トラフィックのモニタリング結果に基づき，スクリーン間の協調制御によりタイミングを調整する方式を採用した．

本方式を適用した VDI 対応監視制御システムの評価システムを開発し，自動再生時のロバスト性，キャプチャ画像の再現性について評価を行った．評価実験の結果，自動再生時のロバスト性については，トラフィックモニタ機能を導入することにより，シナリオの完走率が向上することが確認でき，その結果シナリオの自動再生時のロバスト性の向上が確認できた．キャプチャ画像の再現性についても，準備した評価用シナリオに対する操作に対しては，ほぼ 100%に近い画素一致率が得られていることが確認できた．

以上のように，提案方式は VDI 対応監視制御システムの回帰テストの効率化に対する有効性を確認することができた．

今後の課題としては，1) テスト結果自動確認との組合せによる自動再生失敗箇所の検出方式の確立，2) 評価用のシ

ナリオに基づく評価実験を通じた再生時間のモデル化による作業時間の見積り方式の確立，があげられ，今後はこの点を考慮した形で実用化にむけた開発を継続していく予定である．

参考文献

[1] 日経コンピュータ，日経 Linux，日経 NETWORK（編）：すべてわかる仮想化大全，日経 BP (2014).

[2] 渡部洋司，三田村謙一，工藤謹正：電力系統監視制御システムの技術動向，東芝レビュー，Vol.63, No.4, pp.2–5 (2008).

[3] 菅井尚人，塚原　研，鶴　薫，郡　光則：次世代電力系統制御システム向けプラットフォーム技術，三菱電機技報，Vol.89, No.11, pp.600–604 (2015).

[4] 情報処理推進機構セキュリティセンター：重要インフラの制御システムセキュリティと IT サービス継続に関する調査報告書：IPA 独立行政法人情報処理推進機構，情報処理推進機構（オンライン），入手先 ⟨https://www.ipa.go.jp/security/fy20/reports/ics-sec/⟩（参照 2017-08-10）．

[5] 資源エネルギー庁：電力小売全面自由化—資源エネルギー庁，資源エネルギー庁（オンライン），入手先 ⟨http://www.enecho.meti.go.jp/category/electricity_and_gas/electric/electricity_liberalization/⟩（参照 2017-08-10）．

[6] Richardson, T. and Levine J.: The Remote Framebuffer Protocol, RFC6143 (2011).

[7] 三菱電機：三菱電機公共・エネルギー系統変電システム：電力系統監視制御システム，三菱電機（オンライン），入手先 ⟨http://www.mitsubishielectric.co.jp/business/public/transformation/watch/index.html⟩（参照 2017-08-10）．

[8] 東芝：はじめに：中央給電指令所システム/給電・集中監視制御システム：エネルギーシステムソリューション社：東芝，東芝（オンライン），入手先 ⟨http://www.toshiba.co.jp/tandd/jp/tands/window/products/control/seo.htm⟩（参照 2017-08-10）．

[9] 日立製作所：電力系統監視制御システム：日立，日立製作所（オンライン），入手先 ⟨http://www.hitachi.co.jp/products/infrastructure/product_site/ems/index.html⟩（参照 2017-08-10）．

[10] 本坂洋一，井上信二郎：新プラットフォームを採用した電鉄用監視制御システム，明電時報，Vol.2017, No.2, pp.24–27 (2017).

[11] ASTER テストツール WG：テストツールまるわかりガイド（入門編）Version 1.0.0，ソフトウェアテスト技術振興協会（オンライン）．入手先 ⟨http://aster.or.jp/business/testtool_wg/pdf/Testtool_beginningGuide_Version1.0.0.pdf⟩（参照 2017-08-10）．

[12] Hewlett Packard Enterprise Development: Unified Functional Testing, Hewlett Packard Enterprise Development（オンライン），入手先 ⟨https://saas.hpe.com/ja-jp/software/uft⟩（参照 2017-08-10）．

[13] 日本オラクル：Oracle Functional Testing，日本オラクル（オンライン），入手先 ⟨http://www.oracle.com/technetwork/jp/ats-tech/products/oracle-functional-testing-518227-ja.html⟩（参照 2017-08-10）．

[14] 日本 IBM：IBM — Rational Functional Tester，日本 IBM（オンライン），入手先 ⟨http://www-03.ibm.com/software/products/ja/functional⟩（参照 2017-08-10）．

[15] マイクロフォーカス：マイクロフォーカス製品—テストツール—Silk Test：マイクロフォーカス，マイクロフォーカス（オンライン），入手先 ⟨http://www.microfocus.co.jp/products/silk/silktest/⟩（参照 2017-08-10）．

[16] 日本マイクロソフト：Microsoft Visual Studio ホームページ—Visual Studio，日本マイクロソフト（オンライン），入手先 ⟨https://www.microsoft.com/ja-jp/dev/default.aspx⟩（参照 2017-08-10）．

[17] Selenium Project: Selenium — Web Browser Automation, Selenium Project (online), available from ⟨http://www.seleniumhq.org/⟩ (accessed 2017-08-10).

[18] Timothy Wall: Abbot framework for automated testing of Java GUI components and programs, SourceForge.net (online), available from ⟨http://abbot.sourceforge.net/doc/overview.shtml⟩ (accessed 2017-08-10).

[19] Christian Spannagel and Daniel Herding: Jacareto download — SourceForge.net, SourceForge.net (online), available from ⟨http://sourceforge.net/projects/jacareto/⟩ (accessed 2017-08-10).

[20] Matthew Pekar: Pounder — Java GUI Testing Utility: download — SourceForge.net, SourceForge.net (online), available from ⟨https://sourceforge.net/projects/pounder/⟩ (accessed 2017-08-10).

[21] Tim Waugh: RFB Proxy, cyberelk.net (online), available from ⟨http://cyberelk.net/tim/rfbproxy/⟩ (accessed 2017-08-10).

[22] Constantin Kaplinsky: VNC Reflector: download — SourceForge.net, SourceForge.net (online), available from ⟨http://sourceforge.net/projects/vnc-reflector/⟩ (accessed 2017-08-10).

[23] nagalenoj: Life's Good: FBS files, blogspot.jp (online), available from ⟨http://nagalenoj.blogspot.jp/2010/04/fbs-files.html⟩ (accessed 2017-08-10).

[24] Nickolai Zeldovich: VNCplay: Interactive Session Replay and Performance Measurement, Stanford University (online), available from ⟨http://suif.stanford.edu/vncplay/⟩ (accessed 2017-08-10).

[25] Xen Project: The Xen Project, the powerful open source industry standard for visualization, Xen Project (online), available from ⟨http://www.xenproject.org/⟩ (accessed 2017-08-10).

[26] KVM Project: KVM, KVM Project (online), available from ⟨https://www.linux-kvm.org/page/Main_Page⟩ (accessed 2017-08-10).

[27] 鶴崎真理子，阿倍博信：マルチプラットフォーム対応 GUI 試験省力化方式，2015 年電子情報通信学学会総合大会，D-3-2，p.23 (2015).

[28] David P. Wiggins: XVFB, X.Org Foundation (online), available from ⟨http://www.x.org/releases/X11R7.6/doc/man/man1/Xvfb.1.xhtml⟩ (accessed 2017-08-10).

[29] Runge, K.J.: x11vnc: A VNC server for real X displays, karlrunge.com (online), available from ⟨http://www.karlrunge.com/x11vnc/⟩ (accessed 2017-08-10).

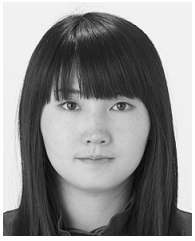
[30] TightVNC Project: TightVNC: VNC-Compatible Free Remote Control/Remote Desktop Software, TightVNC Project (online), available from ⟨http://www.tightvnc.org/⟩ (accessed 2017-08-10).

[31] Amos Waterland: stress project page, Harvard University (online), available from ⟨http://people.seas.harvard.edu/~apw/stress/⟩ (accessed 2017-08-10).

[32] 日本規格協会（編）：JIS ハンドブック色彩 2013，日本規格協会 (2013).

阿倍 博信 （正会員）

1988 年慶応義塾大学理工学部計測工学科卒業，1990 年同大学院理工学研究科修士課程修了，同年三菱電機株式会社入社．以来，グループウェアシステム，マルチメディア応用システムの研究開発に従事．2005 年慶應義塾大学大学院理工学研究科後期博士課程修了．博士（工学）．電子情報通信学会，日本ソフトウェア科学会，映像情報メディア学会，画像電子学会，教育システム情報学会，自動車技術会，計測自動制御学会各会員．



鶴崎 真理子

2010 年九州大学工学部電気情報工学科卒業，2012 年同大学院システム情報科学府修士課程修了，同年三菱電機株式会社入社．以来，マルチメディア応用システムの研究開発に従事．電子情報通信学会会員．

塚原 研

1991 年大阪大学工学部環境工学科卒業．1993 年同大学院工学研究科修士課程修了．同年三菱電機株式会社入社．以来，電力系統監視制御システムの開発に従事．