RMX 開発者用回帰テストツールの開発

長野　薫† 五嶋　研人† 小坂　祐介† 遠山　元道†

† 慶應義塾大学理工学部情報工学科　〒 223–8522 神奈川県横浜市港北区日吉 3-14-1
E-mail: †{nagano,goto,kosaka}@db.ics.keio.ac.jp, ††toyama@ics.keio.ac.jp

あらまし　RMX とは、ルールをあらかじめ定義しておくことで, 動的にデータベースから宛先を取得し, メール配信を行うシステムである. ユーザに品質の高い RMX を提供するために, テストを行うことは非常に重要である. 本論文では, RMX の開発を用意にし, 品質を向上させるために, テストツールの提案と実装を行った. 本ツールでは, GUI 上から RMX のテストケースの管理を行うことができ, また, GUI 上から操作が行えるため, RMX の開発者が簡単にテストの実行・追加・削除を行うことが可能である.

キーワード　RMX, 回帰テスト, メール, テストツール

1. はじめに

Rule-based e-Mail eXchange(RMX) は, 管理者があらかじめ定義したルールに基づき, データベースから動的にメールを転送するメール転送エージェントである [3] [4]. 従来のメールアドレスはアカウント名とドメイン名から構成される 2 次元的なものであるのに対し, RMX ではメールアドレスは, 配送ルール, パラメータ, ドメイン名の 3 つから構成される 3 次元的なものである. しかしながら, 現在の RMX には回帰テストを作成・実行するためのツールやテストケースは存在しておらず, RMX 開発時の回帰テストはこれまで, 実際にメールを送信してみた上での目視・手作業の確認のみであった. 常にデグレードすることなく質の高いシステムをユーザに提供し続けるためには, 回帰テストは時間や手間がかかっても必須であり [1],RMX の品質を向上させるべく, 予期せぬバグやエラーの発生を防ぎながら開発を進めていくためには, 細部まで検査を行える回帰テストを実行する必要があると考えた. そこで, 本論文では, RMX の開発者がスムーズな開発を行うために, RMX 専用の回帰テストツールの作成を提案する. 本ツールは, GUI による操作によって RMX のテストケース作成・実行・管理を行うツールである. データベースにあらかじめ登録されている期待される出力 (テスト解答) とテスト実行によって得られた出力を比較することで検査を行い, 複数出力されるメール全てを比較することも可能である. 以下, 本稿の構成を示す. まず 2 章で RMX の概要を述べる. 次に, 3 章では関連研究について, 4 章では現在の RMX 開発における問題点を示す. 次に, 5 章ではツールの概要を述べ, 6 章では評価について述べ, 最後に 7 章で結論を示す.

2. RMX

Rule-based e-Mail eXchange(RMX) は, 慶應義塾大学が提案している電子メールの配信方法である. RMX 形式のメールアドレスを用いることで, 配送先のメールアドレスを動的に取得し, メールを転送することが可能である. 類似した研究としては, Michael Kasso らの Semantic Email Addressing(SEA) [5] [6] がある. これは, 宛先を動的に取得してメールを送信するという点で RMX と類似しているが, システム独自の GUI を使わないとメールの送信が行えない. RMX では, 特定の GUI に頼ることなく, 直感的に理解することが可能なアドレスでのメール送信が可能である.

RMX のメールアドレス記述には, 関数形式と自然形式の 2 種類が存在する. 以下が関数形式の記述方法である.

> <RMX のメール配信先指定 (関数形式)>:=
> < 配送ルール名 >< パラメータ >@< サブドメイン名 >.< ドメイン名 >

次に, 関数形式を用いたアドレスの例を示す.

> faculty{science}@keio.rmv.jp

この例では, サブドメインは “keio” であるので, “keio.properties” という設定ファイルを参照する. また, “faculty” の部分は, “science” をパラメータに持つ配送ルールを参照する. この際, 配送ルールは設定ファイル内にパラメータ付 SQL 文で記述する. 上記の情報を用いることで, データベースから問い合わせ処理を行い, 配送先アドレスを取得し, それらのアドレスに対してメールの配送を行う. この RMX の一連の処理の流れを図示したものが以下の図 1 である.

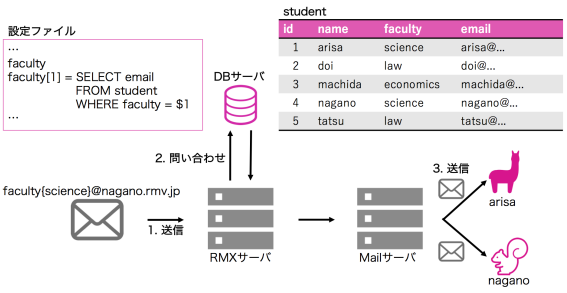
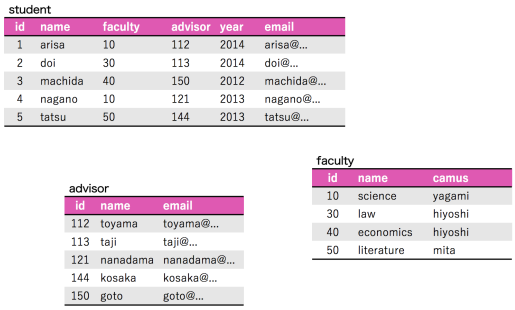


図 1　RMX の処理の流れ

この図からもわかるように, faculty{science}@keio.rmv.jp にメールを送信すると, 理工学部に所属している学生に対してメー

ルが送信されることとなる．以下, 2.1 章でサブドメイン設定ファイル, 2.2 章で配送ルールについての詳細を述べ, 2.3 章で複数の配送ルールの組み合わせについて概説する．説明に際し, 図 2 の例題を使用する．この図は, 大学が保持している学生の情報であり, student テーブルは学生の個人情報を, advisor テーブルは担当教員の個人情報を, faculty テーブルは学部の一覧を示している．それぞれのテーブルはエンティティとなっている．student テーブルの advisor 属性は, advisor テーブルの主キーに対応する外部キーであり, student テーブルの faculty 属性は, faculty テーブルの主キーに対応する外部キーである．



student

id	name	faculty	advisor	year	email
1	arisa	10	112	2014	arisa@...
2	doi	30	113	2014	doi@...
3	machida	40	150	2012	machida@...
4	nagano	10	121	2013	nagano@...
5	tatsu	50	144	2013	tatsu@...

advisor

id	name	email
112	toyama	toyama@...
113	taji	taji@...
121	nanadama	nanadama@...
144	kosaka	kosaka@...
150	goto	goto@...

faculty

id	name	camus
10	science	yagami
30	law	hiyoshi
40	economics	hiyoshi
50	literature	mita

図 2　大学の保持する学生情報 DB

2.1　設定ファイル

設定ファイル内では, データベースの管理及び配送ルールの定義を行う．設定ファイルはサブドメイン名.properties という名前を使用する．例えば, “keio” というサブドメインに対しては “keio.properties” という設定ファイルを置く．サブドメインが “keio” のメールが送られてくると, “keio.properties” という設定ファイルに記述されているデータベースが参照され, そこに書かれているルールが使用される．データベース管理のために設定ファイルに記述する項目は以下の通りである．

- dbDriver = データベースドライバ
- dbURL =接続データベースの URL
- dbID = データベースのユーザ ID
- dbPassword = ユーザ ID に対するパスワード

2.2　配送ルール

配送ルールとは, 配送範囲記述とそれに基づき送信先アドレスを取得するクエリを関連付けるルールである．配送ルールは設定ファイル内に, 下記の様に定義する．

```
配送ルール名
配送ルール名 [パラメータ数]: 送信先アドレスを得るためのクエリ
```

送信先アドレスを得るためのクエリは SQL により記述する．配送ルール名に対応するクエリの ‘$1’ で示されるプレースホルダーにパラメータを挿入し, データベース問い合わせを行うことで送信先メールアドレスの集合を取得する．このように, ユーザは完結な記述で配送範囲を記述することが可能となる．以下に, 配送ルールの定義例を示す．

keio.properties

```
student
student[1] = SELECT email FROM student WHERE year = $1
student[2] = SELECT email FROM student
WHERE year >= $1 AND year <= $2

advisor
advisor[1] = SELECT s.email FROM student s, advisor a
WHERE s.advisor = a.id AND a.name = $1

faculty
faculty[1] = SELECT s.email FROM student s, faculty f
WHERE s.faculty = f.id AND f.name = $1

campus campus[1] = SELECT s.email FROM student s, faculty f
WHERE s.faculty = f.id AND f.campus = $1
```

上記の例は図 2 に対応している．クエリにより図 2 のような既存のデータベースに対して配送ルールを定義できる．この例では student ルールは全ての入学年度の学生に対してそれぞれメールを送信することが可能である．例えば, “student{2014}@keio.rmv.jp” にメールが送信されると, RMX サーバは “keio.properties” の設定ファイルから student ルールを参照する．このメールアドレスはパラメータが 1 つのため, 一番上の配送ルールが呼び出され, $1 部分にパラメータの 2014 が代入される．そして, 取得されたアドレスへメールが送信される．この例では, 2014 年に入学した学生 (arisa, doi) へメールが送信される．

2.3　複数の配送ルールの組み合わせ (関数形式)

RMX では, 配送ルール間に演算子を用いることで, ユーザに対してより詳細な配送範囲の指定を可能にする．

2.3.1　積　集　合

$Syntax : < name_1 >< par_1 > .\cdots. < name_n >< par_n > @ < subdomain > . < domain >$

$Semantics : name_1(par_1) \cap ... \cap name_n(par_n)$

“.” は, 複数の配送ルールを使用したい時に用いられる演算子である．指定された複数の配送ルールの結果の積集合を取り, その積集合により得られた結果に対してメールを配送する．

例: student{2013-2014}.faculty{science}@keio.rmv.jp

2.3.2　和　集　合

$Syntax : < name_1 >< par_1 > +...+ < name_n >< par_n > @ < subdomain > . < domain >$

$Semantics : name_1(par_1) \cup ... \cup name_n(par_n)$

“+” は, 論理和によって複数の配送ルールを指定する際に用いられる. 各パラメータをそれぞれの配送ルールのクエリに代入し, 得られた結果の和集合から配送を行う. また, 配送ルール間の積集合をとる “.” と和集合をとる “+” が同時に使用された場合, “.” の方が優先される.

例: student{2013-2014}.faculty{science}+student{2016}
.faculty{science}@keio.rmv.jp

例えば上記の様なメールアドレスの場合, student ルールと faculty ルールの積集合をそれぞれ取った後, 2 つの和集合をとる. つまりこの場合, 2013 年, 2014 年, 2016 年に理工学部に入学した学生にメールを送信する.

これらの演算子は組み合わせて使用することもでき, ユーザが配送範囲を指定する際の手助けを行う. メールアドレスは以下の形式をとる.

$ListOpe := -$
$UnionOpe := +$
$RuleOpe := .|+|-$
$Arg := string|integer$
$Para := Arg|ArgListOpePara$
$ParaList := Para|ParaUnionOpeParaList$
$Exp := ruleParaList$
$ExpList := Exp|ExpRuleOpeExpList$
$Address := ExpList@subdomain.domain$

3. 関 連 研 究

3.1 RMX の関連研究

RMX のメーリングリストとしての使用方法に焦点を当て, RMX と同様にメーリングリストシステムとして挙動する, 既存のメール配信システムとの比較を行う. 今回の比較対象は, blaynmail [7], Benchmark Email [8], Access Mail [9] の 3 つである. これらは全て, 企業がマーケティングメールを送信するために開発されたシステムである. 以下の表は RMX とそれぞれの機能を比較した表である.

表 1　RMX と既存ツールの比較

	RMX	blaynmail	ベンチマークイーメール	アクセス E メール
グループの登録	不要	必要	必要	必要
既存の DB の利用	O	X	X	X
複数テーブルからの情報抽出	O	X	X	X
専用エディタ	X	O	O	O

上記より, RMX の特徴として以下 3 点があげられる.

（1） グループごとの宛先の登録が不要

（2） 既存のデータベースを直接利用することが可能

（3） SQL を用いるため宛先の記述力が高い

1 点目は, メールアドレスからその場で複数の送信先を決定するためグループごとの登録が不要という点である. Benchmark Email では, 送信するグループごとに登録が必要だが, RMX では不要である. 2 点目は, 既存のデータベースを直接利用することが可能であり, メール配信のために改めて宛先情報を登録する必要がない点である. blaynmail では, 登録が必要だが, RMX では既存のデータベースを用いることができる. 3 点目は, SQL で記述可能な範囲であれば, RMX では配送ルールを落ちいてどのような組み合わせの宛先も抽出可能であるという点である. 上記の点から, RMX は既存システムと比べ, 複数のテーブルをまたぐような広い範囲での宛先指定が可能で有るといえる. これは, 既存のメール配信システムと大きく異る点だと考えられる.

3.2 回帰テストの関連研究

回帰テストはシステムのデグレードを防ぎ常にユーザに品質の高いシステムを提供し続けるためには必要不可欠であり，古くから研究されてきた．代表的な汎用性の高い回帰テストツールとして，Selenium [13]，Apache Jmeter [14]，Jenkins [15] などがあげられる．Selenium は，ブラウザ上で動作する Web アプリのテストを自動化するツールである．テスト中に不具合が生じた際に，不具合発生時の環境や手順が再現できるキャプチャリプレイがあることが特徴である．しかし，RMX はウェブアプリケーションではないため，Selenium を用いて回帰テストをすることはできない．Apache Jmeter は，Web サーバなどに対してのパフォーマンスの測定や，高負荷状態の不可テストを行うことが可能である．負荷テストだけでなく，機能テストのツールとして用いることも可能である．テスト結果をグラフで表示したり，結果をメールで通知することも可能である．しかし Jmeter では，RMX メールサーバの負荷テストなどを行うことは可能だが，RMX における入出力はメールであるため，Jmeter を用いて回帰テストを行うことは困難である．Jenkins は自動化ツールのひとつである．Jenkins に回帰テストを自動的に行うプラグインを紐付けることでテストツールとして用いることも可能である．ブラウザ上で手軽に利用できることから，継続的インテグレーションを実施する企業の多くで用いられている．Jenkins において RMX の回帰テストを自動で行うことはできるが，従来の RMX の回帰テスト法と同じで実際にメールを送信してみてメールボックスをひとつひとつチェックすることになる．アウトプットの比較が手動という点では従来のテスト方法との違いは小さく，Jenkins を用いるメリットは少ないといえるだろう.

4. RMX 開発における問題点

RMX は, 慶應義塾大学遠山研究室で開発されている．不具合の修正によるコードの変更や新機能の実装などを行う場合は手作業でメールを送信し, 目視で受信したメールの比較を行う必要があった．そのため 目視のテストによる「見落とし」「失敗」「不正」といった人的要因によるリスクが存在していた．ま

た, 回帰テストのために複数の受信及び確認ができるメールアドレスを用意する必要があった. 一見同じようなメールが同じ受信アドレス集合に送付されている様に見えても, 件名や本文までもが全く同一であるか確認することは手間であった. 実際に, 遠山研究室にて開発をすすめる際も自身が実装した機能がこれまで使用可能だった機能に影響を与えていないかどうかの確認は, 複数のフリーメールアドレスを用意して何通りもメールを送信して確かめていた.

デグレードを防ぐためには, コード変更前に通過したテストケースを再実行する回帰テストが必要である. 一般的に Java を用いた開発では, Java で開発されたぷおグラムにおいて単体テストの自動化を行うためのフレームワーク JUnit や TestNG などを利用した回帰テストを行う [10], [11]. RMX 開発においても従来のツールを導入し回帰テストを行うという方法も考えられたが, RMX はメール配信システムであるため単純に生成物の比較を行うことが難しいため独自のテストツールを開発する必要があった. 本論文で提案するツールでは, RMX の生成物であるメールを, 宛先集合や件名, 本文, エラーメールについて検査することが可能である. また GUI を用いた操作により, テストの実行・管理を行い, 開発者がテストする際の負担を軽減している.

5. RMX 回帰テストツール

5.1 概　　要

本ツールは Java を用いて実装された Java アプリケーションである. 本ツールでは, RMX のテストケースの実行及び管理を GUI 上から行うことができる.

5.2 システムの構成

本ツールは【RegressionTestRun】, 【TestCaseList】, 【TagList】, 【Preference】の 4 つのタブから成り立っている. 図 3, 図 4, 図 5, 図 6 はそれぞれを表している.

5.2.1 テスト実行タブ

【RegressionTestRun】タブでは, テストの実行及び結果表示を行う. テストの実行を行いたい場合は画面左のツリーにチェックを入れることで実行するテストケースを選択し, 実行ボタン (Execute) を押す. するとテストが実行され, 画面中央に実行結果が表示される. テスト実行中には進捗バーが現れ, テスト実行の進捗状況がパーセント表示される. また, テストをタグごとに実行できる. 画面右上部にはプルダウンが表示されており, 実行結果ごとに結果表示ができる. 最後に, テスト実行が終了すると, テスト結果が左下に表示される.

また, 画面左のツリーは, 下の階層のすべてにチェックが入ると, その上の階層にもチェックが入るようになっている. 1 つでも未選択のものがあれば, 図 7 のように上階層は赤で未選択であることが示される.

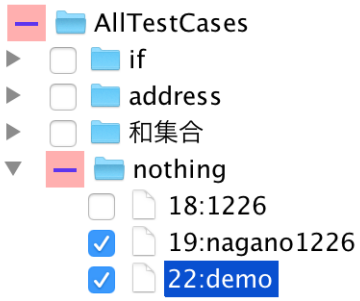


図 7　ツ　リ　ー

5.2.2 テストケース一覧

【TestCaseList】タブでは, 登録されているテストケース一覧の表示及びテストケースの追加・削除ができる. テストケース追加は, 画面下部の追加 (Add) ボタンを押すことで追加画面に遷移に内容入力が可能となる. 追加時には一旦 RMX が実行され, 実行が成功すればテストケース追加確認画面へ, 失敗すればそれでも追加するかどうかの確認画面へと遷移する. テストケース一覧のチェックボックスにチェックを入れて削除 (Delete) ボタンを押すことで不要なテストケースの削除ができる. また各テストケースに関する詳細は, 右の詳細 (Detail) ボタンを押すことで確認できる.

5.2.3 タ グ 一 覧

【TagList】タブでは, 現在登録されているタグ一覧とタグの追加・削除ができる. タグには, address, if, 和集合などが登録されている. タグ名はクリックできるようになっており, そのタグに登録されているテストケース一覧が画面下部に一覧表示される. これに関しても詳細 (Detail) ボタンを押すことで詳細情報を表示できる.

5.2.4 設　　定

【Preference】タブでは, RMX 本体の位置, 設定ファイルやデータベースの位置, 実行結果及び解答の出力先, 解答保持数, ユーザ名の設定などを行う

5.3 処理の流れ

5.3.1 テストケース実行処理の流れ

全体としての処理の流れは図 8 の通りである.

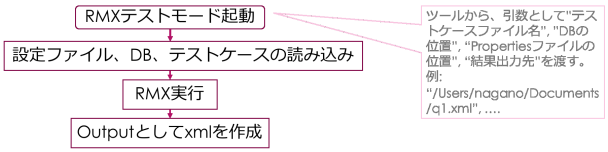


図 8　ツールの処理の流れ

データベースには事前に, テストケースと対応するテスト解答が保存されている. まず, 本ツール上でテスト実行が行われデータベースに接続する. 次に, 登録されている各テストケースに対する最新の解答のみを選択し, テストケースとテスト回答を取得する. そして, 取得したテストケースと解答のそれぞ

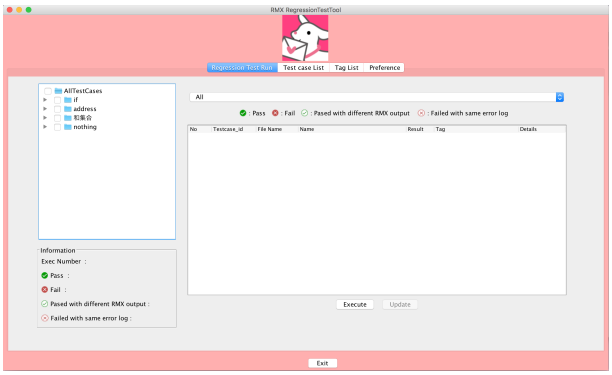


図 3　テスト実行タブ

図 4　テストケース管理タブ

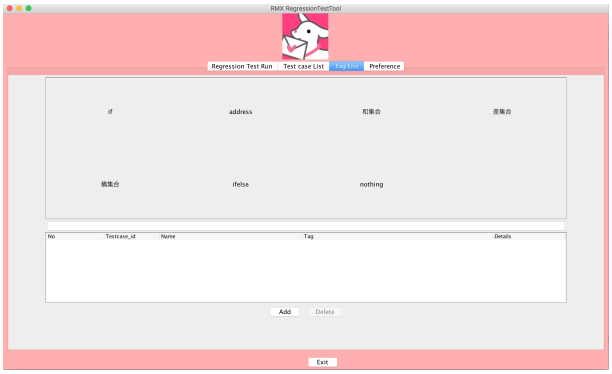


図 5　タグ管理タブ

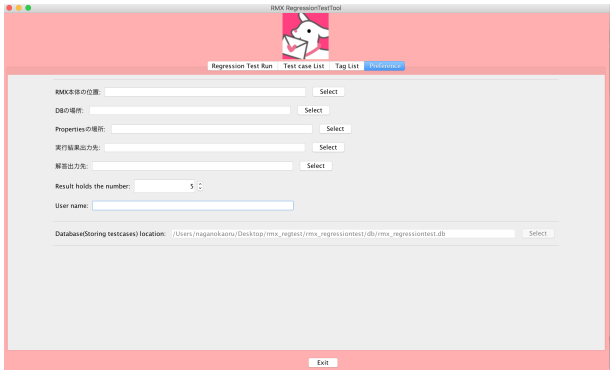


図 6　設 定 タ ブ

れをファイルに書き出し，テストケースの実行結果と解答ファイルの内容を比較する (図 9)．本来 RMX では起動後メール受信を待機するが，テスト実行時にはメールの代わりに，本来ならば RM が受信したメールから読み取る情報をまとめて記述してある，図 10 のようなテストケース XML ファイルを読み込む．<to>タグ内にメールの宛先アドレスが，<sender>タグ内に送信者メールアドレスが，<subject>内に件名が，<body>内に本文が記載されており，これらの情報全てが<mail>タグに囲われている．

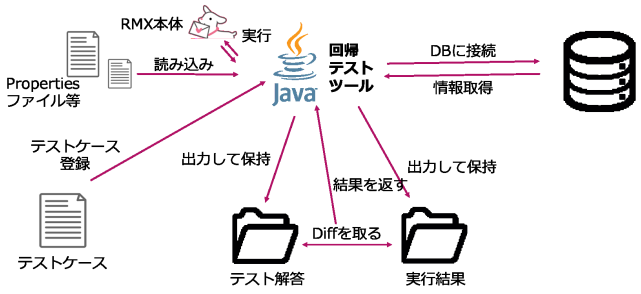


図 9　テストツールの処理の流れ

```
<?xml version="1.0" encoding="UTF-8"?>
    <mail>
        <to>dept{ics}+dept{chemistry}@nagano.rmv.jp</to>
        <sender>nagano@db.ics.keio.ac.jp</sender>
        <subject>件名を入力</subject>
        <body>ここには本文が入ります。
            テスト

</body>
    </mail>
```

図 10　テストケースの一例

5.3.2　テストケース追加処理の流れ

本ツール上で追加ボタンが押されると図 11 のようなテストケース追加画面が現れる．そこで必要情報を入力し追加ボタンを押すと，データベースに接続しテストケースと解答が追加が行われる．削除ボタンを押すと確認画面が現れ，ok を押すと追加時同様にデータベースに接続し該当するテストケース及び解答が削除される (図 12)．

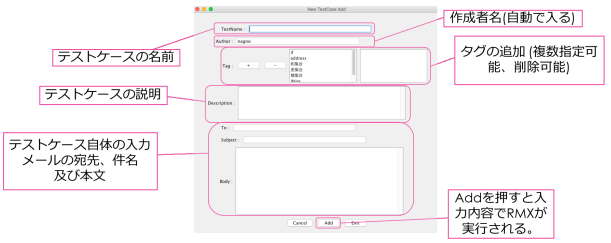


図 11　テストケースの追加

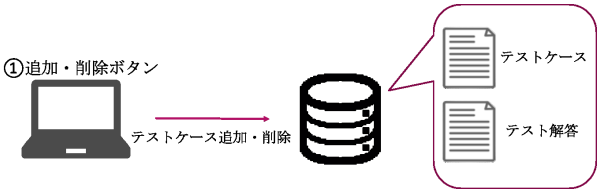


図 12　テストケースの追加

5.3.3　差分表示

以下で, 5.3.1 で述べたテストケース実行結果とテスト解答の比較について詳細を述べる. 全体としての処理の流れは図 13 の通りである.

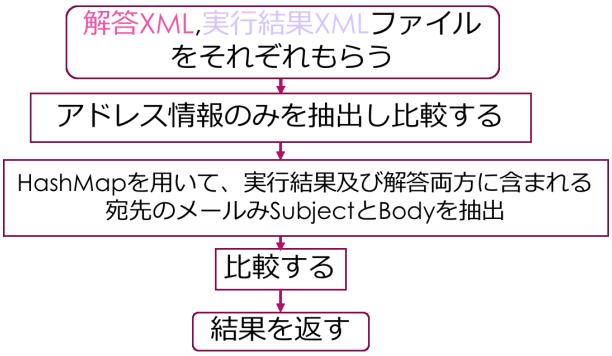


図 13　差分表示の処理の流れ

本ツールでは, 実行結果と解答との間で生成されるメールの宛先集合及び個々のメールの件名, 本文をそれぞれ差分を取り結果を表示している. 図 14 は比較箇所を表示する画面である. 画面左上部のボックスには, 実行結果及びテスト解答の両方でメールの宛先となったアドレスが一覧表示されている. それぞれのアドレスをクリックすると画面中央及び右部に実行結果, テスト解答でそれぞれ生成されたメールの件名及び本文が表示される. 実行結果とテスト解答で異なる件名または本文が生成された場合は異なる箇所が赤くハイライトされ, 画面下部に差分の行数と内容が詳細表示される. また左中央部にはテスト解答の宛先集合には含まれるが実行結果の宛先集合には含まれていないアドレス集合が, 左下部のボックスにはテスト解答の宛先集合には含まれていないが実行結果の宛先集合には含まれているアドレス集合がそれぞれ表示される.

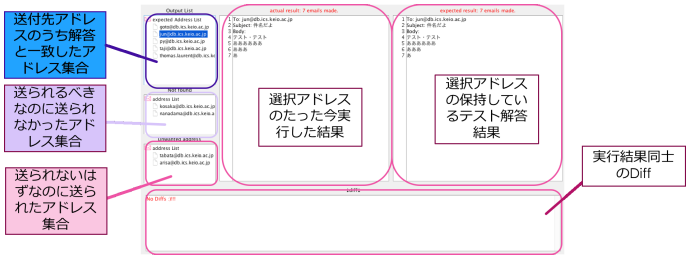


図 14　比較画面

6.　評　　価

本研究で提案するテストツールの有用性を評価するため, 本論文で提案するツール (以下 “ツール” と表記) と, 従来の RMX テスト方法 (以下 “従来のテスト法” と表記), RMX テストモードをコマンドラインで起動した場合 (以下 “コマンドラインテスト起動” と表記) についてそれぞれ全ての差分のあるメールを発見するまでにかかる時間を比較する. なお, 従来のテスト法とは RMX テストモードを用いず実際にメールを送信してみて, 配送されたメールすべてについて差分がないか確認することをいう. また, コマンドラインテスト起動とは, コマンドラインにて RMX テストモードを起動し, 回帰テストを行い, 作成された XML ファイルをテスト解答 XML ファイルと目視で比較し差分がないか確認することをいう. 実験についての詳細は以下で述べる.

RMX の回帰テストにおいて考えられる差分は以下の 2 通りである.

- 配送先メールアドレス (宛先) 集合の不一致
- 各配送先メールアドレス (宛先) に対する件名または本文の不一致

そこで, 本実験では, RMX 開発者が, 2 のケースですべての上記パターンの差分を発見するまでにかかる時間を比較することでツールの有用性を示す. なお, 本実験では遠山研究室の RMX 開発者 6 人にそれぞれツール, 従来のテスト法及びコマンドラインテスト起動の場合について実行させ, その時間の計測した. なお, ツールの場合は実行ボタンを押してから, 従来のテスト法の場合はメールを送信してから, コマンドラインテスト起動の場合は実行直後から, それぞれすべての差分を発見するまでの時間を計測するものとする. なお, 全ての 2 の実験パターンにおいてテスト解答の配送先アドレスは 10 件とする.

表 2　実験のパターン

パターン	配送先アドレスの差分の数	本文に差分のあるメールの件数
1	0	0
2	+3	0
3	-3	0
4	0	5
5	+4	3
6	-4	3

なお表 2 の 2 列目における “+” はテスト解答よりも配送先アドレスが多い場合を示し, “-” は少ない場合を示している.

図 15　実験結果のまとめ

student

id	name	faculty	dept	email	year	grade	advisor	gender
1	arisa	10	110	arsa@	2016	2	101	female
2	doi	30	310	doi@	2014	4	301	male
3	machida	40	420	machida@	2015	3	405	male
4	nagano	10	150	nagano@	2013	4	108	female
5	tatsu	40	420	tatsy@	2015	3	405	male
…	…	…	…	…	…	…	…	

advisor

id	name	email
101	shu	shu@
108	tbt	tbt@
301	kotani	kotani@
…	…	…

dept

id	name
110	ics
150	chemistry
…	…

faculty

id	name	campus
10	science	yagami
30	law	mita
40	medicine	shinanomachi
…		

図 16　評価実験のデータベース

```
campus
campus[1] = SELECT s.email FROM student s, faculty f WHERE s.faculty = f.id AND f.campus = '$1';

grade
grade[1] = SELECT s.email from student s WHERE s.grade = '$1';

faculty
faculty[1] = SELECT s.email FROM student s, faculty f WHERE WHERE s.faculty = f.id AND f.name = '$1';
faculty[3] = SELECT s.email FROM student s, faculty f WHERE WHERE s.faculty = f.id AND f.name = '$1' or '$2' or '$3';

gender
gender[1] = SELECT s.email FROM student s WHERE s.gender = '$1';
```

図 17　評価実験で用いるプロパティファイル

なお，本実験で用いるデータベース及びプロパティファイルはそれぞれは図 16，図 17 の通りであり，student テーブルに追加されている学生の件数は 30 人とする．つまり，従来のテスト法において確認するメールボックスの数は最大で 30 である．

なお，被験者が申告してきた報告していた差分の件数が間違っていた場合，その旨を記録した上で再度正しい結果を報告するまでの時間を計測する．

実験結果は図 15 である．各パターンについて，実験者 6 人それぞれが差分検出までにかかった時間の平均をグラフに示した．この図からもわかるように，全ての実験パターンについてどの被験者についてもツールを用いることで，差分検出までの時間を約 1/8 程度に短縮できることがわかる．本実験では，メールを受信する可能性のあるメールアドレスは student テーブルにある 30 件のみであったが，よりメールを受信する可能性のあるアドレスの件数が増えればツールの有用性はより示される．また，コマンドラインテスト起動では，従来のテスト手法よりは時間が削減できたものの，生成された XML ファイルを目視で確認することから，ツールを用いる場合よりは時間がかかった．

7. 結　　論

本研究では，RMX の開発者向け専用回帰テストツールを提案し，実装を行った．メール配信システムである RMX の特徴を捉え，開発者が開発しやすくなるようなツールの作成を行った．テストケース作成の際も，説明文を記述できるようにする

ことで仕様を明確にし, 現在の仕様がどうなっているか等も他者がひと目で理解できるようにした. 開発途中でも頻繁にテストすることが可能であり, 問題を早期発見できるようにすることが RMX の品質向上につながると考える.

文　　献

[1] Adriaan Labuschagne, Laura Inozemtseva, Reid Holmes. “Measuring the cost of regression testing in practice: a study of Java projects using continuous integration”, ESEC/FSE 2017.

[2] F.I. Vokolos, P.G. Frankl. “Empirical evaluation of the textual differencing regression testing technique”. ICSM, 1998.

[3] 高畑 理, 藤沼健太郎, 石橋玲, 遠山元道. “Magic Mirror Mailing: 個人情報データベースを利用する柔軟なメイル配送システム” 情報処理学会データベースシステム研究報告, Pages 123–128, July 2001.

[4] Kim Hanki, Sang-Gyu Shin, Motomichi Toyama. “A Rule-Based Mailing System for an Organization”, International Workshop on INformation Processing over Evolving Networks, June 2006.

[5] Michael Kasso, Charles Petrie, Lee-Ming Zen, Michael Genesereth. “Semantic email addressing: sending email to people, not strings”, In Proc. AAAI 2006 Fall Symposium on Integrating Reasoning into Everyday Applications.

[6] Michael Kasso, Charles Petrie, Lee-Ming Zen, Michael Genesereth. “Semantic email addressing: The Semantic Web Killer App?”, Internet Computing 13(1), pp. 48-55, IEEE, 2009

[7] “blaynmail”, http://blaynmail.jp/

[8] “Benchmark Email”, http://www.benchmarkemail.com/jp/

[9] “Access Mail”, http://www.accessmail.jp/

[10] JUnit: http://junit.org

[11] TestNG: http://testng.org/doc/index.html

[12] L.J. White, V.Narayanswarmy, T.Friedman. “Test Manager: A regression testing tool”, Proc. Conf. on Software Maintenance, 1993.

[13] Selenium. http://www.seleniumhq.org/.

[14] Apahe Jmeter. http://jmeter.apache.org/.

[15] Jenkins. http://jenkins-ci.org/.