社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

サマータイム制度によるソフトウェア開発への影響調査

林　　純一† 　肥後　芳樹† 　柗本　真佑† 　楠本　真二†

† 大阪大学大学院情報科学研究科
〒 565-0871 大阪府吹田市山田丘 1-5
E-mail: †{j-hayasi,higo,shinsuke,kusumoto}@ist.osaka-u.ac.jp

あらまし　一部の国や地域では，夏季を中心に標準時刻を早めるサマータイム制度が実施されている．そのため，ソフトウェアにおいて日時を扱う際にサマータイム制度を考慮する必要がある．ただし，例えばサマータイム制度を導入していない地域向けのソフトウェアでは考慮していない場合もある．そのようなソフトウェアにおいても，利用者からの要望などによってサマータイム制度に対応するようにソースコードが変更される場合も考えられる．また，サマータイム制度による影響の調査は，社会学的な視点では行われているが，ソフトウェア工学的な視点では行われていない．そこで本研究では，オープンソースソフトウェアの開発履歴をもとに，サマータイム制度に関するソースコードの変更が行われた時期やその変更を行った開発者の居住地などを調査し，さらにそれらの変更をいくつかのパターンに分類した．

キーワード　サマータイム制度，ソースコード変更，地理的解析

1. ま え が き

一部の国や地域では，夏季を中心に標準時刻を早めるサマータイム制度が実施されている．サマータイム制度は主に日中の経済活動を促進する目的で行われており，社会に対して様々な影響を与えていることが報告されている [1]〜[4]．

日本でもサマータイム制度を導入するか否かの議論が 2018 年に行われた．日本は 2020 年の東京オリンピックを控えており，競技が日中の暑い時間帯に実施されるのを避けるために，サマータイム制度の導入が検討された [5]．これに対して情報法制研究所が，日本国内で利用されている電子機器はサマータイム制度に対応していないものが多く，それらをサマータイム制度の導入までに更新しなければならないが，オリンピック開催までの約 2 年間では対応が間に合わないと反対した [6]．最終的に日本は 2020 年までにサマータイム制度を導入することを断念した [7]．この議論は，ソフトウェアにおけるサマータイム制度への対応が難しいことを示唆するものであり，サマータイム制度がソフトウェア開発に与える影響を調査する必要性が存在すると考える．しかしながら，ソフトウェア工学の視点からサマータイム制度の影響を調査した研究は確認できなかった．

ソフトウェアにおいて日時を扱う際は，一般にサマータイム制度を考慮しなければならない．一般的なプログラミング言語では，日時の情報を複数の方法で取り扱えるため，開発者が実装する取扱方法の特徴を理解した上で実装しなければ，欠陥が発生する可能性があると考える．このことから，サマータイム制度に関連する欠陥を修正するために，ソースコードの変更が行われているのではないかと考える．また，そのようなソースコードの変更には何らかのパターンが存在すると考える．

ソースコードの変更に関する研究は多数行われている [8]〜[10]．著者らは，ソースコードの変更履歴を学習して，修正漏れが発生しているコード片を自動で検出するツールを開発した [9]．Meng らは，ソースコードの変更を学習して，自動的に最適な箇所を変更するツールを開発した [8]．Fluri らは，ソースコードの変更を抽出するための，木構造の差分を求めるアルゴリズムを提案している [10]．

本研究では，サマータイム制度のソフトウェア開発における影響を調査するために，オープンソースソフトウェアの開発履歴をもとに，サマータイム制度に関連するソースコードの変更が行われた時期や，その変更を行った開発者の居住地などを調査した．また，サマータイム制度に関連するソースコードの変更についていくつかのパターンを示した．

本研究の主な貢献は以下の通りである．

- 過去に作成されたソフトウェアほど，サマータイム制度に関する修正が行われていることを明らかにした．
- プログラミング言語に関わらず，日時の取り扱いにおいてサマータイム制度を考慮する必要があることを明らかにした．
- Java で書かれたソフトウェアのサマータイム制度に関する修正について，ソースコードの典型的な変更のパターンを示した．

2. サマータイム制度

サマータイム制度は，1 年のうち日照時間の長い春から秋にかけて，標準時刻を 1 時間早める制度である．この制度は，午

後の日照時間を長くすることによって経済活動を促進する目的で，2018 年現在約 4 割の国で実施されている [11]．例えばアメリカ合衆国では，一部の地域を除いて，3 月第 2 日曜に標準時刻を 1 時間早め，11 月第 1 日曜に元に戻すことになっている [12]．

2.1 社会への影響

サマータイム制度は社会に対して以下のような影響を与えている．

- 経済の活性化 [1]
- 消費電力のピーク需要の減少 [2]
- 交通事故の増加 [3]
- 急性心筋梗塞のリスクの増加 [4]

このように，サマータイム制度が社会に与える影響についての研究は，社会学的な視点から多数行われている．

2.2 ソフトウェア開発への影響

一般に，プログラミング言語において日時を扱う方法は複数存在する．例えば Java の標準ライブラリには，タイムゾーンを考慮して日時を扱うことができる java.time.ZonedDateTime クラスと，タイムゾーンの情報を扱わない java.time.LocalDateTime クラスがある．ソフトウェアの開発者はこれらのクラスの違いを認識した上で所望の処理を実装する必要があり，このような違いは欠陥が発生する要因の一つになっていると考えられる．

このように，サマータイム制度はソフトウェア開発に対して何らかの影響を及ぼしていると著者らは考える．しかし，ソフトウェア工学の視点からサマータイム制度の影響を調査した文献は，著者らが確認した限り存在しなかった．そこで本研究では，ソフトウェア開発へのサマータイム制度の影響を，オープンソースソフトウェアの開発履歴をもとに調査する．

本論文では，サマータイム制度に関連するソースコードの修正を「サマータイム修正」と呼ぶことにする．

2.3 Research Questions

サマータイム制度のソフトウェア開発に対する影響の調査を行うにあたり，以下の 5 つの Research Question を設定した．

RQ1. サマータイム修正はどの程度行われているか．

RQ2. サマータイム修正されたプロジェクトにはどんな特徴があるか．

RQ3. サマータイム修正はいつ行われ，その修正がソフトウェアに適用されるまでに要した時間はどれくらいか．

RQ4. サマータイム修正を行った開発者の在住地はどこか．

RQ5. サマータイム修正においてどのようなソースコードの変更が行われているか．

3. 調 査 方 法

3.1 RQ1 の調査方法

オープンソースソフトウェアに対して行われたサマータイム修正を収集するために，GitHub のキーワード検索機能を用いた．“daylight saving” と “summer time” のキーワードで検索し，既にマージされているプルリクエストを収集した．“daylight saving” はアメリカ英語であり，“daylight saving time” と “daylight saving*s* time” の表記揺れを同時に検索するために採用した．“summer time” はイギリス英語であり，こちらの表現を使っている開発者が行った修正も収集できるようにするために採用した．

サマータイム修正が行われたリポジトリ数の推移を調査するために，サマータイム修正を含むリポジトリの作成日時を GitHub が提供している REST API を利用して収集した．また，GitHub で公開されているすべてのリポジトリとの傾向を比較するために，GH Archive(注1) を利用して，各年に作成されたリポジトリの数を集計した．

3.2 RQ2 の調査方法

RQ1 で収集したサマータイム修正を含むリポジトリについて，次の項目を調査した．

- 最も使用されているプログラミング言語
- ドメイン

最も使用されているプログラミング言語は，GitHub が提供している REST API を利用して収集した．1 つのリポジトリにおいて複数の言語が用いられている場合は，そのうち最も使用されている言語を，そのリポジトリにおける最も使用されている言語とする．ドメインは Borges らが文献 [13] で定義した以下の 6 つとし，筆者らが各リポジトリの説明文や README ファイルを参照して分類した．

Application software エンドユーザに機能を提供するシステム

System software 他のシステムへのサービスやインフラを提供するシステム

Web libraries and frameworks Web に関連するライブラリやフレームワーク

Non-web libraries and frameworks Web に関連しないライブラリやフレームワーク

Software tools ソフトウェア開発における種々の作業を支援するシステム

Documentation ドキュメントやチュートリアル，ソースコード例などのリポジトリ

なお，Borges らの調査によってドメインが既に分類されているリポジトリについては，そのドメインをリポジトリのドメインとした．

3.3 RQ3 の調査方法

サマータイム修正が行われた日時を調査するために，RQ1 で収集した各プルリクエストが作成された日時とマージされた日時を，GitHub の REST API で収集した．また，サマータイム修正が行われてから実際のソフトウェアに適用されるまでに要した日数を調査するために，収集した日時から，各プルリクエストが作成されてからマージされるまでに掛かった時間を計算した．

3.4 RQ4 の調査方法

RQ1 で収集した各プルリクエストにおいてコミットを行った開発者について，各開発者が GitHub 上で公開しているメー

(注1)：https://www.gharchive.org/

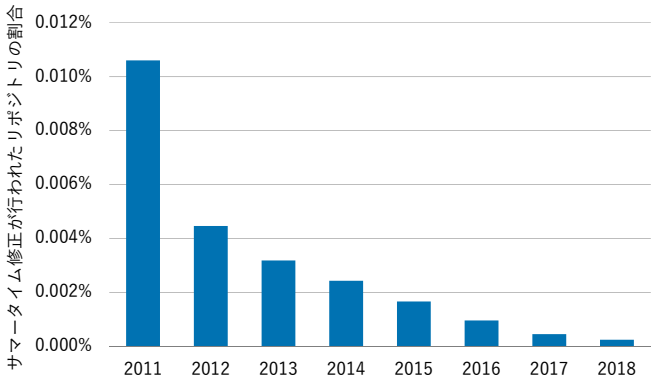


図 1　サマータイム修正が行われたリポジトリの作成時期

ルアドレスのトップレベルドメインをもとに，開発者の居住地を判別した．トップレベルドメインは必ずしも特定の国や地域を示すわけではないが，調査の正確性と容易性の観点からこの方法で調査を行った．したがって，公開されているメールアドレスのトップレベルドメインが国別コードトップレベルドメインのものではない開発者は，本 RQ の調査対象としていない．

3.5　RQ5 の調査方法

RQ1 で収集したプルリクエストのうち，最も使用されている言語が Java であるリポジトリに対するプルリクエストについて，そのプルリクエストで行われたソースコードの変更を筆者らが目視で確認し，変更のパターンを分類した．

4.　調 査 結 果

4.1　RQ1 の調査結果

図 1 に，2011 年から 2018 年までに作成されたリポジトリのうち，サマータイム修正が行われた割合の年毎の推移を示す．2011 年に作成されたリポジトリの割合が最も多く，以後減少傾向にあることが分かる．また，その割合も，最も多い 2011 年で 0.010%であり，全体的にサマータイム修正が行われるリポジトリが少ないことが分かる．

4.2　RQ2 の調査結果

表 1 の左 2 列に，サマータイム修正が行われた各リポジトリにおいて最も使われている言語を集計したものを示す．GitHub 全体の傾向と比較するために，表 1 の右 2 列には GitHut プロジェクトの調査結果 [14] を記載している．左側にある言語は HTML を除きすべて右側にもあるため，サマータイム修正が行われるリポジトリの言語には偏りがないことが分かる．また，割合の数値を見ても，ほとんどの言語が同程度の割合であった．

表 1　リポジトリで最も使われている言語（上位 10 言語）

調査対象リポジトリ		全リポジトリ [14]	
言語	割合（%）	言語	割合（%）
Python	20	JavaScript	21
JavaScript	16	Python	15
Java	10	Java	11
Ruby	6	PHP	8
C++	6	C++	8
PHP	6	C#	6
C	5	Shell	4
HTML	4	Go	4
C#	4	Ruby	4
Go	2	C	4

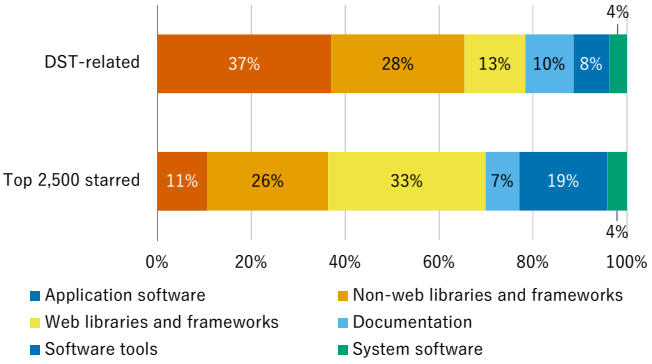
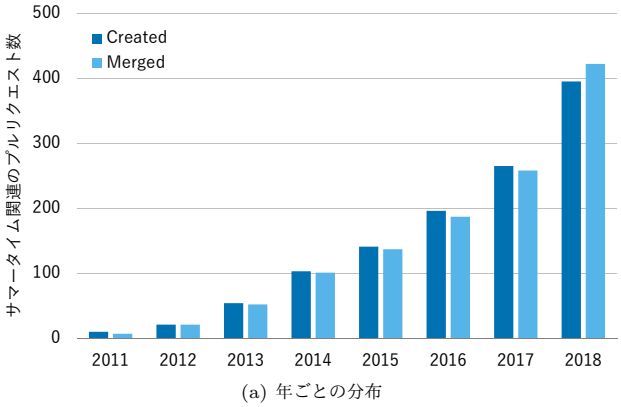
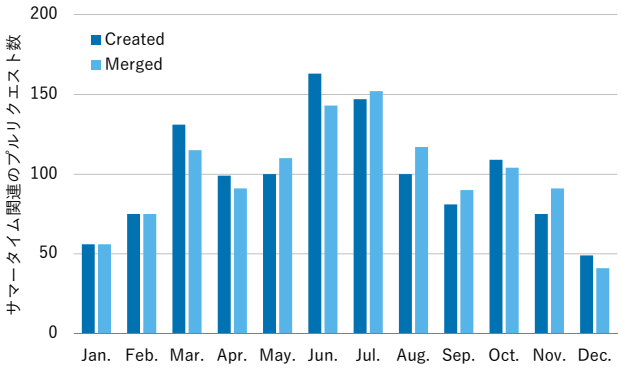


図 2　サマータイム修正が行われたリポジトリのドメイン



(a) 年ごとの分布



(b) 月ごとの分布

図 3　サマータイム修正時期の分布

図 2 の上の棒グラフに，サマータイム修正が行われたリポジトリのドメインを筆者らが分類した結果を示す．なお，図 2 の下の棒グラフは，Borges らが分類した GitHub のスター数上位 2,500 リポジトリのドメインの分布 [13] を，比較のために記載したものである．スター数上位 2,500 リポジトリのドメイン分布と比べ，アプリケーションや Web ライブラリの割合が多くなっている．

4.3　RQ3 の調査結果

サマータイム修正のプルリクエストが作成された時期とマージされた時期の分布を図 3 に示す．図 3(a) は年ごとの分布を，

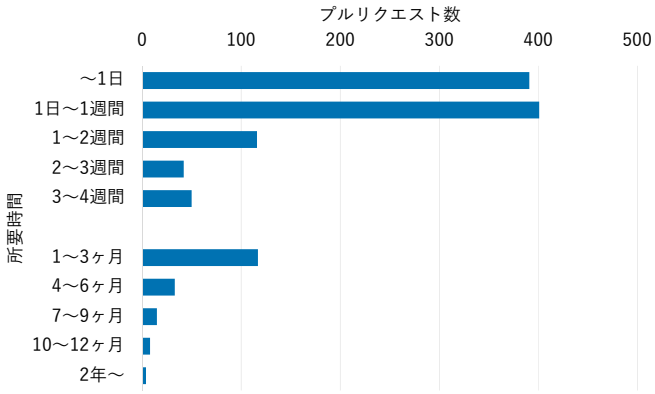


図 4　サマータイム修正が行われてからマージされるまでの所要時間

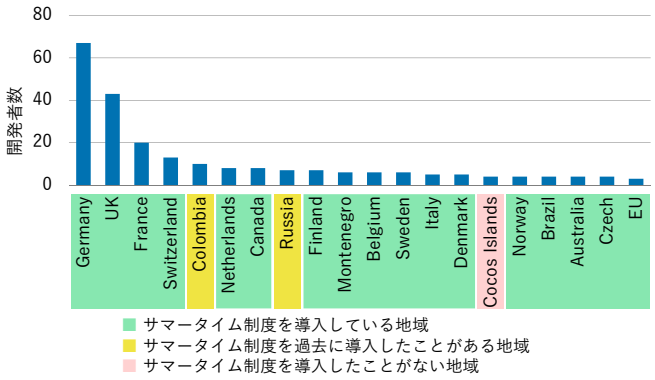


図 5　サマータイム修正に関わった開発者の居住地（上位 20 ヶ国）

図 3(b) は月ごとの分布を示している．これらの図から，サマータイム修正のプルリクエストについて以下の傾向が見られた．

- プルリクエスト作成数は年々増加している．
- 夏季のプルリクエスト作成数が他の季節に比べて多い．

図 4 に，サマータイム修正のプルリクエストが作成されてからマージされるまでに掛かった時間の分布を示す．この図から，ほとんどのプルリクエストは，作成されてから 1 ヶ月以内にマージされていることがわかる．

4.4　RQ4 の調査結果

図 5 に，サマータイム修正に関わった開発者のうち国別コードトップレベルドメインのメールアドレスを公開している開発者について，そのトップレベルドメインから居住地を判別し，集計した結果を示す．なお，サマータイム修正に関わった開発者の総数は 1,210 人で，そのうち国別コードトップレベルドメインのメールアドレスを公開していた開発者は 274 人（約 22%）であった．

図 5 において，背景が緑色である地域は，その地域でサマータイム制度が 2018 年末の時点で実施されていることを示す．背景が黄色である地域は，その地域で過去にサマータイム制度を導入していた時期があるが，現在は実施していないことを示す．背景が赤色である地域は，これまでにサマータイム制度の実施経験がないことを示す．

この調査の結果から，ほとんどのサマータイム修正はサマータイム制度が実施されている地域に住む開発者によって行われていることが分かった．

```
- public static DateTimeFormatter getDateTimeFormatterForDataset(TimeSpec
                                                                 timeSpec) {
+ public static DateTimeFormatter getDateTimeFormatterForDataset(TimeSpec
                                                       timeSpec, DateTimeZone zone) {
    String pattern = null;
    TimeUnit unit = timeSpec.getDataGranularity().getUnit();
    switch (unit) {
      case DAYS:
        pattern = DAY_FORMAT;
        break;
      case MINUTES:
        pattern = MINUTE_FORMAT;
        break;
      case HOURS:
      default:
        pattern = HOUR_FORMAT;
        break;
    }
-   DateTimeFormatter dateTimeFormatter = DateTimeFormat.forPattern(pattern);
+   DateTimeFormatter dateTimeFormatter =
                            DateTimeFormat.forPattern(pattern).withZone(zone);
    return dateTimeFormatter;
  }
```

図 6　新たにタイムゾーンを取り扱う処理を追加する変更の例（APACHE/INCUBATOR-PINOT リポジトリにおける変更(注2)）

```
  private Date getEndDate(Product prod, Date startTime) {
    int interval = maxDevLifeDays;
    String prodExp = prod.getAttributeValue("expires_after");
    if (prodExp != null &&  Integer.parseInt(prodExp) < maxDevLifeDays) {
      interval = Integer.parseInt(prodExp);
    }
-   return 1000 * 60 * 60 * 24 * interval;
+   Calendar cal = Calendar.getInstance();
+   cal.setTime(startTime);
+   cal.add(Calendar.DAY_OF_YEAR, interval);
+
+   return cal.getTime();
  }
```

図 7　日時の計算に java.util.Calendar クラスを使用する変更の例（CANDLEPIN/CANDLEPIN リポジトリにおける変更(注3)）

4.5　RQ5 の調査結果

筆者らがソースコードの変更を調査した結果，以下のような変更が行われていることが分かった．

- 新たにタイムゾーンを取り扱う処理を追加
- 日時の計算に java.util.Calendar クラスを使用
- タイムゾーンを取り扱えるクラスで日時の情報を保持

図 6–8 に，これらの変更の例を示す．これらの図において，背景が緑色の行はその変更で追加された行を示し，背景が赤色の行はその変更で削除された行を示している．

1 つ目の変更は，明示的にタイムゾーンの取り扱いをしていなかったために，実行環境によって時刻が変わってしまう欠陥に対処するものである．図 6 の変更では，メソッドにタイムゾーンの情報を示す org.joda.time.DateTimeZone クラスの引数を追加し，その引数を org.joda.time.format.DateTimeFormatter クラスのオブジェクトにタイムゾーンの情報を渡すことようにしている．この変更により，org.joda.time.format.DateTimeFormatter クラスのオブジェクトが適切なタイムゾーンで日時を扱えるようになっている．

2 つ目の変更は，日時の計算において定数値を用いていたのを，Java の標準ライブラリを利用して計算させるようにしたものである．サマータイム制度を実施している地域において 1 日の長さは，通常時刻とサマータイムの切替時に変わる

(注2)：https://github.com/apache/incubator-pinot/pull/992/files
(注3)：https://github.com/candlepin/candlepin/pull/1082/files

```
- Date date = null;
+ ZonedDateTime date = null;
  …… 中略 ……
  try {
-   date = new SimpleDateFormat(DATE_PATTERN_WITH_TZ).parse(calendarValue);
- } catch (ParseException fpe2) {
-   date = new SimpleDateFormat(DATE_PATTERN).parse(calendarValue);
+   date = ZonedDateTime.parse(zonedValue, formatterTz);
+ } catch (DateTimeParseException tzException) {
+   try {
+     date = ZonedDateTime.parse(zonedValue, formatterTzMs);
+   } catch (DateTimeParseException regularFormatException) {
+     // A ZonedDateTime object cannot be creating by parsing directly
                                                    a pattern without zone
+     LocalDateTime localDateTime = LocalDateTime.parse(zonedValue, formatter);
+     date = ZonedDateTime.of(localDateTime, ZoneId.systemDefault());
+   }
  }
  …… 中略 ……
  if (date != null) {
-   calendar = Calendar.getInstance();
-   calendar.setTime(date);
+   zonedDateTime = date;
  }
```

図 8　タイムゾーンを取り扱えるクラスで日時の情報を保持する変更の例（ECLIPSE/SMARTHOME リポジトリにおける変更(注4)）

ため一定ではない．この違いを考慮するために，既にサマータイム制度を考慮する実装になっている Java の標準ライブラリを利用するように変更している．図 7 の変更では，1 日を常に $1000 \times 60 \times 60 \times 24$ ミリ秒として計算していたところを，java.util.Calendar クラスを用いて計算するようにしている．この変更により，通常時刻とサマータイムの切り替え時に 1 日の長さが変わることに対処できるようになっている．

3 つ目の変更は，日時の管理を，タイムゾーンの情報を取り扱えないクラスを用いて行っていたところを，タイムゾーンの情報を取り扱えるクラスを用いるようにしたものである．図 8 の変更では，タイムゾーンの情報を保持しない(注5)java.util.Date クラスで管理していたところを，タイムゾーンの情報を保持する java.time.ZonedDateTime クラスで管理するように変更している．この変更により，サマータイム制度に関する情報を含むタイムゾーンの情報を扱えるようになっている．

5. 考　　察

5.1 RQ1 の考察

4.1 節で述べた結果から，最近作られたリポジトリよりも昔に作られたリポジトリのほうがサマータイム修正が行われていることが分かった．これは，時間の経過によってリポジトリの利用者が増加し，利用者からの要望やバグ報告が増加するためであると考える．

5.2 RQ2 の考察

4.2 節で述べた結果から，割合の数値に若干の差異はあるものの，サマータイム修正が行われたリポジトリと GitHub 上の全リポジトリで用いられている言語の傾向はほとんど同じであった．このことから，どの言語においても日時を取り扱う際にサマータイム制度を考慮して実装を行う必要があると考えられる．

5.3 RQ3 の考察

4.3 節で述べた結果から，サマータイム修正の件数は年々増加し，多くの地域でサマータイム制度が実施される 3 月から 10 月までの件数が多いことが分かった．これは，日時を扱うリポジトリの総数が時間の経過とともに増加し，それに伴ってサマータイム修正の件数も増加しているのではないかと考える．また，サマータイム制度の実施期間中に修正の件数が多くなるのは，その期間にサマータイム制度に起因する不具合が現れ，利用者から報告されるためであると考えられる．

さらに，開発者がサマータイム修正を行ってから，その修正が実際のソフトウェアに適用されるまでに要した日数は，ほとんどのプルリクエストが 3 ヶ月以内であった．これらの結果から，サマータイムが実施されている期間中に，開発者によってサマータイム修正が行われ，実際のソフトウェアに適用されることが多いといえる．

5.4 RQ4 の考察

4.4 節で述べた結果から，サマータイム制度が実施されていない地域の開発者がサマータイム修正に関わっていない傾向があることが分かった．これは，サマータイム制度が実施されていない地域ではサマータイム制度を考慮する必要性が小さく，そのような地域に住む開発者はあまりサマータイム修正に関わっていないためであると考える．

5.5 RQ5 の考察

4.5 節で述べた結果から，ソフトウェアのリリース当初におけるサマータイム制度への対応が不十分であり，後に修正が行われているプロジェクトが存在することが分かった．これは，1 つの言語において日時を取り扱う方法が複数存在し，サマータイム制度を考慮しない方法で実装していたとしても，実装した時期によっては欠陥が現れず，サマータイム制度の実施機関中になって初めて欠陥が発見され，開発者による修正が行われるためであると考える．

6. 妥当性の脅威

RQ1 の調査において，サマータイム修正を収集するために，GitHub のキーワード検索機能を利用した．この方法では，キーワードがプルリクエストで使われていないがサマータイム修正が行われているプルリクエストが収集できていないことや，検索結果には現れるがサマータイム制度に関する修正が含まれていないプルリクエストがあることが考えられる．これらを考慮して精確に調査した場合，異なる傾向が得られる可能性がある．

RQ2 の調査において，リポジトリのドメインを，リポジトリの説明文や README ファイルを目視確認することで判別した．そのため，適切でないドメインに分類されているリポジトリが存在する可能性がある．

RQ3 の調査において，プルリクエスト作成数について，夏季の方が他の季節よりも多いと結論づけたが，統計的に有意であるかどうかの検定までは行っていない．この調査は，単にサマータイム修正の件数の推移を調べたものであるため，このような評価で十分であると考える．

(注4)：https://github.com/eclipse/smarthome/pull/4259/files

(注5)：より正確には，タイムゾーンのオフセットを返すメソッドは存在するが，そのメソッドは非推奨（deprecated）とされている [15].

RQ4 の調査において，調査の対象とする開発者を，公開しているメールアドレスが国別コードトップレベルドメインのものである者に限定した．4.4 節で述べた通り，そのような開発者は全体の 1/4 に満たなかったため，開発者全体の傾向を明らかにできたとはいえない．特に，サマータイム制度が実施されているアメリカ合衆国に住む開発者は多いと考えられるが，アメリカ合衆国の国別コードトップレベルドメインである .us のメールアドレスを使用している開発者が少ないため，調査対象に偏りが生じている．今回の調査で対象としなかった開発者についても在住地を調査することにより，結果が変わる可能性がある．

7. あとがき

本研究では，サマータイム制度のソフトウェア開発に対する影響の調査として，サマータイム制度に関連した修正が行われたリポジトリの特徴，修正の時期，修正を行った開発者の居住地などを調査した．調査の結果，サマータイム制度が関連する欠陥などに対応するための修正が年々増加しており，特にサマータイム制度の実施期間中に修正が行われることが分かった．また，Java で開発されているプロジェクトの修正内容を精査したところ，サマータイム制度に関連して行われた修正にはいくつかのパターンが見受けられた．

今後の課題としては，サマータイム制度に関連するバグについての調査や，ソースコードの変更に関する調査を Java 以外の言語に対して行うことなどが挙げられる．

謝辞　本研究は，日本学術振興会科学研究費補助金基盤研究（B）（課題番号：17H01725）の助成を得て行われた．

文　献

[1] M.B. Aries and G.R. Newsham, “Effect of daylight saving time on lighting energy use: A literature review,” Energy Policy, vol.36, no.6, pp.1858–1866, 2008.
[2] P. Hancevic and D. Margulis, “Daylight saving time and energy consumption: The case of Argentina,” MPRA Paper 80481, University Library of Munich, Germany, 2016.
[3] T. Monk, “Traffic accident increases as a possible indicant of desynchronosis,” Chronobiologia, vol.7, no.4, pp.527–529, 1980.
[4] M.R. Jiddou, M. Pica, J. Boura, L. Qu, and B.A. Franklin, “Incidence of myocardial infarction with shifts to and from daylight savings time,” The American Journal of Cardiology, vol.111, no.5, pp.631–635, 2013.
[5] “【東京五輪】酷暑対策でサマータイム導入へ　秋の臨時国会で議員立法　３１、３２年限定,” Aug. 2018. 産経ニュース. https://www.sankei.com/politics/news/180806/plt1808060002-n1.html
[6] 清嶋直樹，“「サマータイム、間に合わない」　IT 識者ら導入反対,” Sept. 2018. 日本経済新聞. https://www.nikkei.com/article/DGXMZO34948480U8A900C1000000/
[7] “五輪サマータイム断念表明　自民、法案提出困難,” Nov. 2018. 日本経済新聞. https://www.nikkei.com/article/DGXMZO38025970R21C18A1000000/
[8] N. Meng, M. Kim, and K. McKinley, “Lase: Locating and applying systematic edits by learning from examples,” 2013 35th International Conference on Software Engineering, ICSE 2013 - Proceedings, pp.502–511, Oct. 2013.
[9] 肥後芳樹，楠本真二，“コード修正履歴情報を用いた修正漏れの自動検出,” 情報処理学会論文誌，vol.54，no.5，pp.1686–1696，May 2013.
[10] B. Fluri, M. Wuersch, M. PInzger, and H. Gall, “Change distilling:tree differencing for fine-grained source code change extraction,” IEEE Transactions on Software Engineering, vol.33, no.11, pp.725–743, Nov. 2007.
[11] S. Thorsen, “Daylight Saving Time – DST – Summer Time”. https://www.timeanddate.com/time/dst/
[12] M. Fitzpatrick, Daylight Saving Time, Connecticut General Assembly, Dec. 2017.
[13] H. Borges, A. Hora, and M.T. Valente, “Understanding the factors that impact the popularity of github repositories,” 2016 IEEE International Conference on Software Maintenance and Evolution (ICSME), pp.334–344, Oct. 2016.
[14] C. Zapponi, “Github Language Stats,” Oct. 2018. https://madnight.github.io/githut/#/pushes/2018/3
[15] Oracle, “Date (Java SE 11 & JDK 11),” Sept. 2018. https://docs.oracle.com/en/java/javase/11/docs/api/java.base/java/util/Date.html