2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)

Impacts of Daylight Saving Time on Software Development

Junichi Hayashi, Yoshiki Higo, Shinsuke Matsumoto and Shinji Kusumoto
Graduate School of Information Science and Technology, Osaka University, Japan
{j-hayasi, higo, shinsuke, kusumoto}@ist.osaka-u.ac.jp

***Abstract*—Daylight saving time (DST) is observed in many countries and regions. DST is not considered on some software systems at the beginning of their developments, for example, software systems developed in regions where DST is not observed. However, such systems may have to consider DST at the requests of their users. Before now, there has been no study about the impacts of DST on software development. In this paper, we study the impacts of DST on software development by mining the repositories on GitHub. We analyze the date when the code related to DST is changed, and we analyze the regions where the developers applied the changes live. Furthermore, we classify the changes into some patterns.**

***Index Terms*—Daylight Saving Time, Code Change Analysis, Geographical Analysis**

I. Introduction

Many countries and regions observe daylight saving time (in short, DST). DST is the practice of advancing clocks during summer months, so that evening daylight lasts longer while sacrificing normal sunrise times. In 2018, there was much controversy over whether to introduce DST before the 2020 Tokyo Olympic games in Japan because introducing DST can reduce physical burden for athletes by playing their games in the early morning. However, as a result, Japan ended up not introducing DST because the Japanese business community strongly opposed introducing DST. The community loudly said that introducing DST requires enormous modifications on software systems used in Japanese society and it is not realistic to finish such modifications before the 2020 Tokyo Olympic games.

The controversy gave the authors an idea that treating DST in software systems may be a difficult task. We also consider that there may be some coding patterns for practically treating DST in software systems. In other words, the authors think that DST probably has some impacts on software development and maintenance. The authors found several research studies from a sociological perspective [1]–[4] while we were not able to find any research studies from a software engineering perspective.

In this research, we investigated source code changes due to DST by using Git repositories of OSS projects. The main contributions of this paper are as follows.

- This is the first investigation on DST from a software engineering perspective.
- We found that more mature projects have more modifications related to DST.
- Modifications related to DST do not depend on programming languages. Software projects of any programming languages have DST modifications.
- We found some change patterns in DST modifications in Java projects.

II. Daylight Saving Time

Daylight saving time (DST) is a seasonal time change measure where clocks are set ahead of standard time during part of the year, usually by 1 hour. As DST starts, the sun rises and sets later, on the clock, than the day before. Today, about 40% of countries and regions use DST to make better use of daylight and to conserve energy [5]. For example, most of the United States begin DST on the second Sunday in March and revert to the standard time on the first Sunday in November [6].

A. Impacts on Society

Introducing DST has the following impacts on our society.

- DST makes an energetic economy and society [1].
- DST reduces the peak demand for electricity [2].
- DST increases the number of traffic accidents [3].
- DST increases the risks of acute cardiac infarction [4].

As shown in the above, there are several research studies from a sociological perspective.

B. Impacts on Software Systems

There are multiple ways to treat date and time in the source code. For example, the standard library of Java includes classes `java.time.ZonedDateTime` and `java.time.LocalDateTime`. The former one considers time zone, but the latter one does not. If a developer does not know the differences between the two classes, he/she may induce bugs in the source code. The authors consider that introducing DST also has some impacts on software development and maintenance. However, we did not find any research studies on DST from a software engineering perspective. In this research, we investigate the impacts of DST on software systems by using Git repositories of OSS projects. Hereafter, we call code changes related to DST *DST-changes*.

III. Research Questions

We set up the following research questions in our research.

RQ1. How many projects conduct DST-changes?
RQ2. What kinds of software systems include DST-changes?
RQ3. When are DST-changes conducted?

2574-3864/19/$31.00 ©2019 IEEE
DOI 10.1109/MSR.2019.00076

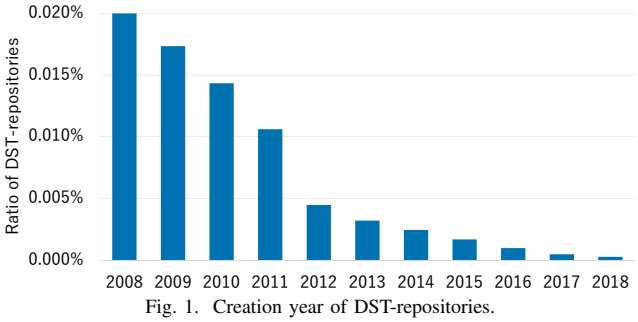

Fig. 1. Creation year of DST-repositories.

RQ4. Which countries and regions do developers of DST-changes live in?

RQ5. How is code changed in DST-changes?

The following subsections of this section describe the methodology, results and discussion for research questions above.

A. RQ1. How many projects conduct DST-changes?

1) Methodology: We focus on pull requests (in short, PRs) in Git repositories on GitHub in order to analyze DST-changes conducted in OSS projects. The PRs that we focused on satisfy the following both two conditions:

- PRs that are related to DST, and
- PRs that have already been merged.

We search for such PRs by keywords which mean DST, "daylight saving" and "summer time". These words are used in the United States and the United Kingdom. The keyword "daylight saving" is chosen to correct the orthographical variants of DST (i.e., "daylight saving time" or "daylight savings time") at a time. We call repositories including DST-related PRs *DST-repositories*.

The creation date of a repository is an attribute of the software project. The creation date of each DST-repository was retrieved by using a REST API provided by GitHub.

2) Experimental Results: There are 1,181 DST-related PRs in 969 projects.

Figure 1 shows the distribution of years when the DST-repositories were created. The Y-axis of the figure means the ratio of the DST-repositories created in the year to all repositories created in the same year. The number of all repositories which were created in each year since 2011 was aggregated using the data on GH Archive[1]. Since GH Archive does not have data between 2008 and 2010, the number of all repositories which were created in each year until 2010 was extrapolated based on the article [7] of GitHub's blog.

Many DST-repositories were created in 2008. The ratio of the DST-repositories decreases as time passed. The absolute value of the ratio of the DST-repositories gets decreased year by year.

3) Discussion: The experimental results show that older projects have more DST-changes than the projects started recently. We consider the reasons for the results are that OSS projects acquire new users, that users report bugs related to DST, and that users need the treatments for DST, as time passed.

4) Answer to RQ: 969 projects are conducted DST-changes.

B. RQ2. What kinds of software systems include DST-changes?

1) Methodology: We checked the following two attributes of each DST-repository:

- the most used language in the repository, and
- the application domain of the repository.

The most used language in each repository is retrieved using a GitHub API. If multiple languages are used in a given repository, the most used language in the repository is regarded as the programming language of the repository. The application domains are defined by Zapponi et al. as follows [8]:

- Application software: systems that provide functionalities to end-users, like browsers and text editors.
- System software: systems that provide services and infrastructure to other systems, like operating systems, middleware, servers and databases.
- Web libraries and frameworks.
- Non-web libraries and frameworks.
- Software tools: systems that support software development tasks, like IDEs package managers, and compilers.
- Documentation: repositories with documentation, tutorials, source code examples, and so on.

We classified them into these domains manually by reading the description and a README file of the DST-repositories.

2) Experimental Results: The left part of Table I shows the top 10 languages used in the DST-repositories. For the comparisons with all repositories on GitHub, the right part of Table I represents the ranking of the most used languages for all repositories in GitHub. This ranking is reported by GitHut project [9]. In Table I, most of the languages listed in the left part are listed in the right part. The proportions of each language are almost the same as well. Thus, the tendencies of the languages used in both the DST-repositories and all repositories on GitHub are very similar.

Figure 2 shows the distribution of the application domains of the repositories. The upper bar in the figure shows the distribution of the DST-repositories, and the lower bar shows the distribution of the top 2,500 starred repositories on GitHub

TABLE I
TOP-10 LANGUAGES USED IN DST-REPOSITORIES.

DST-repositories		All repositories [9]	
Language	Proportion (%)	Language	Proportion (%)
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

[1]https://www.gharchive.org/

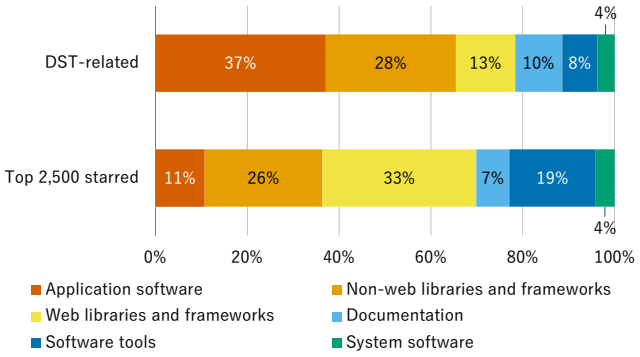

Fig. 2. Application domains of DST-repositories.

according to Borges et al. [8]. Application software and Non-web libraries and frameworks of the DST-repositories are more than those of the top 2,500 starred repositories.

3) Discussion: The experimental results show that the rate of each language that DST-changes conducted are almost the same as the rate of each language in all GitHub projects. Consequently, we consider that DST-changes are necessary for any languages.

4) Answer to RQ: Software systems of Application software and Non-web libraries and frameworks include more than ones of other domains.

C. RQ3. When are DST-changes conducted?

1) Methodology: The creation date of a DST-related PR means when a developer conducted DST-changes on source code. The date merged the PR means when developers adopted the changes. We retrieved those dates by the APIs of GitHub.

2) Experimental Results: Figure 3 shows the number of the DST-related PRs created and merged in every year. The parts of each bar in the figure are painted in different colors to distinguish the year when the repository including the PR was created. Figure 4 shows the number of the DST-related PRs created and merged on every month. Note that each number in Figure 4 is a total of the repositories created or merged on the same months between 2011 and 2018.

The number of created PRs increases year by year as shown in Figure 3. We can see that the DST-related PRs were created and merged in summer more than in other seasons.

3) Discussion: The experimental results show there are two tendencies in DST-changes.

- More DST-changes are conducted in recent years.
- More DST-changes are conducted during March to October, which is almost the same as the period of DST.

We consider the reason for the first tendency is that new repositories treating DST are created every year. In such systems, DST-changes are conducted every year after the repository creation. The reason for the second tendency may be that new bugs are found in operating software systems with DST mode.

The results also show that there are repositories conducted DST-changes constantly every year in spite of the years when

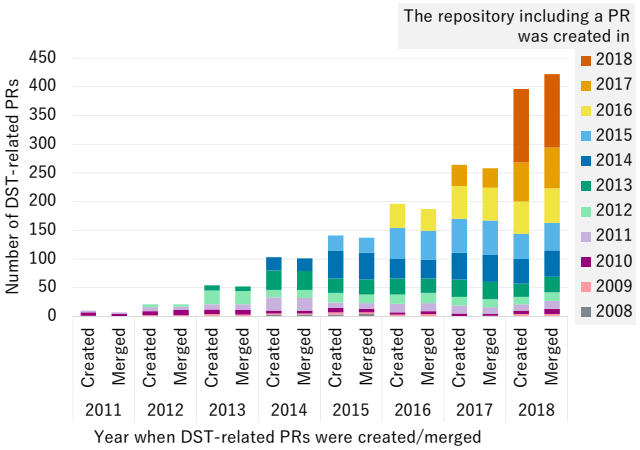

Fig. 3. The number of DST-related PRs which were created in every year. The parts of each bar are painted in different colors to distinguish the year when the repository including the PR was created.

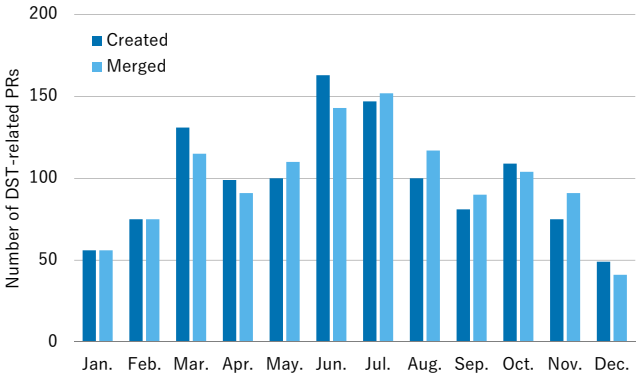

Fig. 4. The number of DST-related PRs which were created every month between 2011 and 2018.

they were created. This is another reason for the second tendency and also leads to the reason why more DST-repositories were created formerly than recently.

4) Answer to RQ: More DST-changes are conducted during the period of DST than other.

D. RQ4. Which countries and regions do developers of DST-changes live in?

1) Methodology: We identified the country or region of each developer that conducted some DST-changes, based on the top-level domains (TLDs) of their public email addresses. Although every TLD does not indicate specific regions, we investigated the email address with country code TLDs that indicate the residential regions of the developer, because such regions can be identified correctly and automatically.

2) Experimental Results: We found that 1,210 developers modified some source code in the DST-related PRs and we identified the residential regions where 274 developers out of them live. Figure 5 shows the top 20 residential regions. In the figure, the regions highlighted in green are observing DST at present. The regions highlighted in yellow has observed DST before, however, they do not observe it at present. The region highlighted in red has never observed DST.

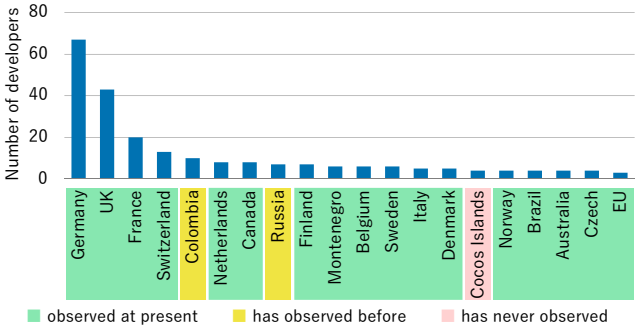

Fig. 5. Top 20 residential regions of the developers that modified some source code in DST-related PRs.

As shown in the figure, most of the developers live in the regions observing DST. In other words, most of DST-changes were conducted by the developers living in the regions observing DST.

3) Discussion: The experimental results show that DST-changes are conducted by developers who live in DST countries or regions. The fact means that developers who live in non-DST countries or regions tend not to conduct DST-changes. Software systems used in non-DST countries and regions do not count for DST, which may be a reason for this tendency.

4) Answer to RQ: Most of developers conducting DST-changes live in countries and regions that observe DST.

E. RQ5. How is code changed in DST-changes?

1) Methodology: We investigated code changes in the DST-related PRs to identify common fixing patterns in the changes. The changes are conducted in Java repositories because we are used to reading Java code.

2) Experimental Results: We found the following patterns of DST-changes from the DST-related Java repositories:

- adding the process that handles time zones, and
- using java.util.Calendar class for the calculation of date and time.

The former change is applied mainly when different times are given according to runtime environments because the time zone was not considered before the change. The latter change is that the calculation of date and time depends on classes of the standard library of Java, instead of using constant values without consideration of DST, e.g., a day is equal to $24 \times 60 \times 60 = 86{,}400$ seconds. Figures 6 and 7 are practical patterns of the changes. In the figures, the lines added in the change are highlighted in green and the lines removed in the change are highlighted in red.

3) Discussion: The experimental results show that there are many projects that had insufficient treatments for DST and then they had received DST-changes. This may be because testing code related to DST is difficult in non-DST season. The existences of multiple ways for treating date and time may be another reason for DST-changes. The code does not consider DST may not be exposed before DST starts.

4) Answer to RQ: There are changes that classes and APIs using management and calculation of date and time make more appropriate.

[2]https://github.com/apache/incubator-pinot/pull/992/files
[3]https://github.com/candlepin/candlepin/pull/1082/files

```
- 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;
  }
```

Fig. 6. Example change adding the process that handles time zones, for APACHE/INCUBATOR-PINOT[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();
  }
```

Fig. 7. Example change using java.util.Calendar class to calculate date and time, for CANDLEPIN/CANDLEPIN[3].

IV. Conclusion

In this research, we investigated the impacts of daylight saving time on software systems. Especially, we focused on features of repositories where changes related to daylight saving time were conducted, date and time of such changes, and residence of developers who conducted such changes. As a result, we found that more changes related to daylight saving time are conducted in recent years and most of such changes are conducted in the period of daylight saving time. We also investigated what kinds of code had been modified due to daylight saving time, and we found that software systems occasionally get releases even they include insufficient treatment of daylight saving time. In the near future, we are going to investigate faults that are related to daylight saving time. We are also going to conduct investigations of code changes for other languages than Java.

Acknowledgment

This work was supported by MEXT/JSPS KAKENHI 17H01725 and 18H03222.

REFERENCES

[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," University Library of Munich, Germany, MPRA Paper 80481, 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] S. Thorsen, "Daylight Saving Time – DST – Summer Time." [Online]. Available: https://www.timeanddate.com/time/dst/

[6] M. Fitzpatrick, *Daylight Saving Time*. Connecticut General Assembly, Dec 2017.

[7] B. Doll, "10 Million Repositories," Dec 2013. [Online]. Available: https://github.blog/2013-12-23-10-million-repositories/

[8] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," in *2016 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Oct 2016, pp. 334–344.

[9] C. Zapponi, "Github Language Stats," Oct 2018. [Online]. Available: https://madnight.github.io/githut/#/pushes/2018/3