

# Characterize Software Release Notes of GitHub Projects: Structure, Writing Style, and Content

---

Jianyu Wu\*, Weiwei Xu\*, Kai Gao\*, Jingyue Li<sup>†</sup>, Minghui Zhou\*<sup>‡</sup>

\*School of Computer Science and School of Software & Microelectronics, Peking University, Beijing, China

\*Key Laboratory of High Confidence Software Technologies, Ministry of Education, Beijing, China

<sup>†</sup>Norwegian University of Science and Technology, Trondheim, Norway

Proceedings of the 30th International Conference on Software Analysis,  
Evolution and Reengineering (SANER '23), pp473-484

# 背景 | リリースノート (RN)

| ソフトウェアのリリースの際にユーザに示す資料またはウェブサイト

- 機能強化, 不具合修正の内容 など

| リリースが早期, 頻繁に行われるようになり, RNを作成するペースは早く, プロセスは困難に

- RNの作成は複雑で労力と時間のかかるプロセス<sup>[1][2]</sup>
- RNのための統一された基準やガイドラインは存在しない<sup>[1]</sup>

[1] L. Moreno, G. Bavota, M. D. Penta, R. Oliveto, A. Marcus, and G. Canfora, "Arena: An approach for the automated generation of release notes," IEEE Transactions on Software Engineering, vol. 43, no. 2, pp. 106–127, 2017.

[2] T. Bi, X. Xia, D. Lo, J. Grundy, and T. Zimmermann, "An empirical study of release note production and usage in practice," IEEE Transactions on Software Engineering, 2020.

# 先行研究 | RNに含まれる情報の理解

| 一般的にRNに含まれる情報を理解するための研究がいくつか行われている<sup>[1][2][3]</sup>

| GitHubのRNに関する問題の調査<sup>[4]</sup>

- 構造が悪く，重要な変更点が埋もれてしまう
- 文体にも問題があり，明快さや理解しやすさが損なわれている

RN構造のベストプラクティス  
RNの文体については未調査のまま

[3] S. L. Abebe, N. Ali, and A. E. Hassan, “An empirical study of software release notes,” Empirical Software Engineering, vol. 21, no. 3, pp. 1107–1142, 2016.

[4] J. Wu, H. He, W. Xiao, K. Gao, and M. Zhou, “Demystifying software release note issues on github,” in 2022 IEEE/ACM 30th International Conference on Program Comprehension (ICPC), 2022, pp. 602–613.

# 研究目的・アプローチ

## 研究目的

- ソフトウェアのドメインやリリースの種類を問わず、さまざまなRNの構造、文体、内容について明らかにする

## アプローチ

- 612件のRNに対してオープンコーディングを行い、RNの構造と文体を調査
- 構造、文体、内容の分布を比較し、異なる種類のリリースがどのようにRNを生成するかを理解

# 調査方法 | データセット

## プロジェクトの選定

- 以下の条件を満たすプロジェクトを選定（955件）
  - 2019年7月1日（3年前）以前に作成，2022年に少なくとも1回のコミットがある
  - 10回以上のリリースが行われている
  - スター数が10k以上
- さらにランダムに選択，除外（233件）
  - 非ソフトウェアプロジェクト，非英語プロジェクトなど

## リリースノートの選定

- リリースタイプの特定にSemVer<sup>[5]</sup>を利用（612件）
  - メジャー：162件，マイナー：227件，パッチ：223件

[5] “Semantic versioning from semver.org,” <https://semver.org/>, 07 2022.

# 調査方法 | 分類方法

## ドメインの分類

- 既存研究<sup>[6]</sup>に基づいて分類
  - 4つのソフトウェアドメインに分類

## 構造, 文体, 内容の分類

- 内容
  - 既存研究<sup>[2]</sup>に基づいて内容を8つに分類
- 構造, 文体
  - 目視による分類と後述のインタビューにより, 4種類, 3種類に分類

[6] 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). IEEE, 2016, pp. 334–344.

# 調査方法 | 技術者インタビュー

- | 構造, 文体に関する分類を検証するために, 技術者2名に半構造化インタビューを実施
  - 2回実施
    - 1回目 「変更の整理に用いる構造」 「変更を説明する方法」
    - 2回目 「本調査で分類した構造と文体について」

# 調査結果 | 構造

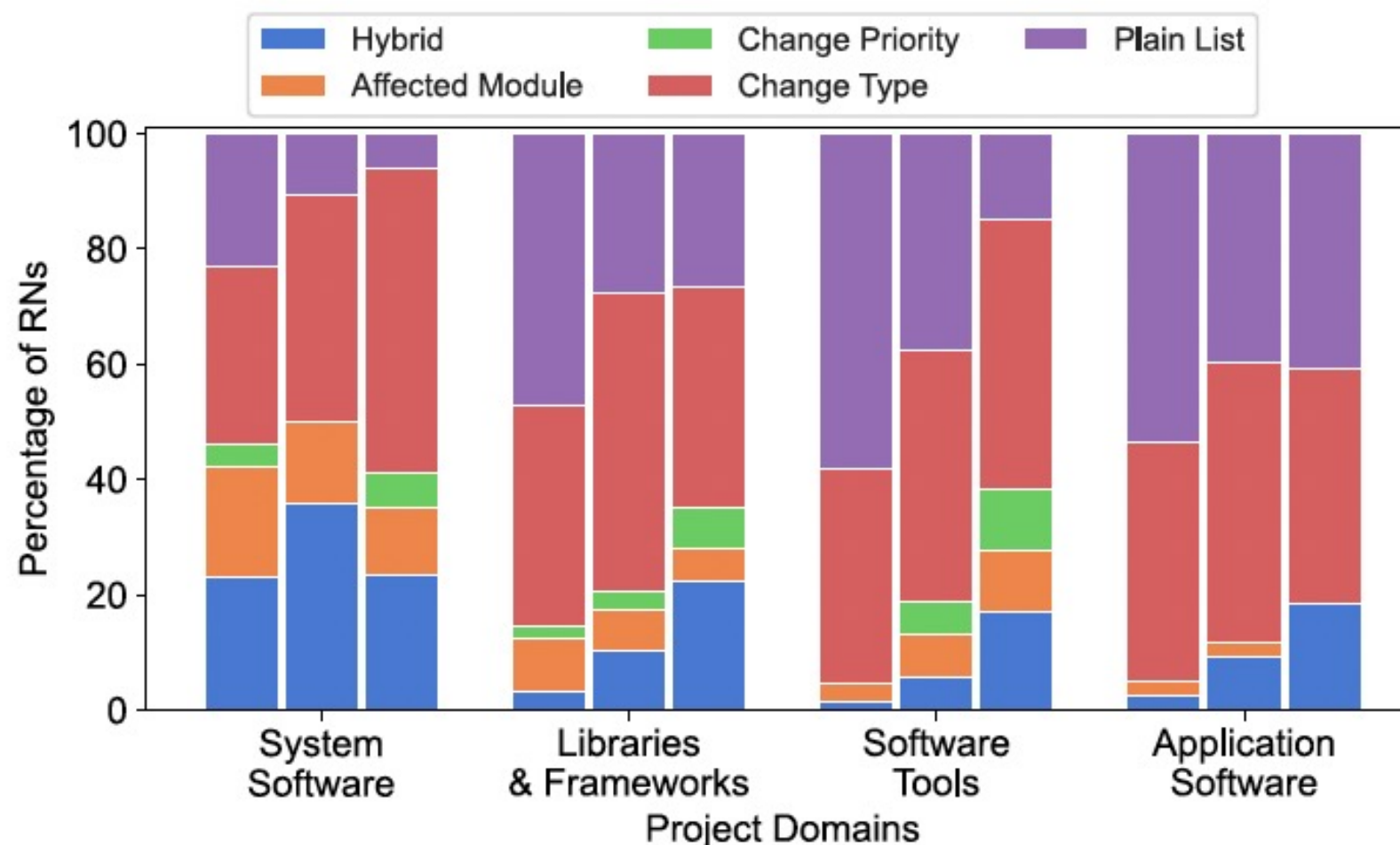
RNの1/3がプレーンなリストとして, 2/3が階層的なリストとして構成されている

階層を作成するために, 変更種類, 影響を受けるモジュール, 変更の優先順位の3つの戦略を採用している

| Strategy                     | Description                                                                                                  | Major. | Minor. | Patch. | Total. |
|------------------------------|--------------------------------------------------------------------------------------------------------------|--------|--------|--------|--------|
| Plain List                   | All changes are presented as a plain list.                                                                   | 23.46% | 30.84% | 48.88% | 35.46% |
| Change Type <sup>a</sup>     | Changes are organized based on their types of content, e.g., New Features, Fixed Bugs, and Breaking Changes. | 62.96% | 58.59% | 42.15% | 53.76% |
| Affected Module <sup>b</sup> | Changes are grouped based on the modules they affect.                                                        | 20.37% | 16.30% | 11.66% | 15.69% |
| Change Priority <sup>c</sup> | Changes are ranked based on their importance perceived by RN producers.                                      | 13.58% | 6.16%  | 2.24%  | 6.70%  |

# 調査結果 | 構造

- すべてのドメインにおいて、変更は主に変更タイプによって整理
- システムソフトウェアのRNは、影響を受けるモジュールによって変更を整理する傾向が強い



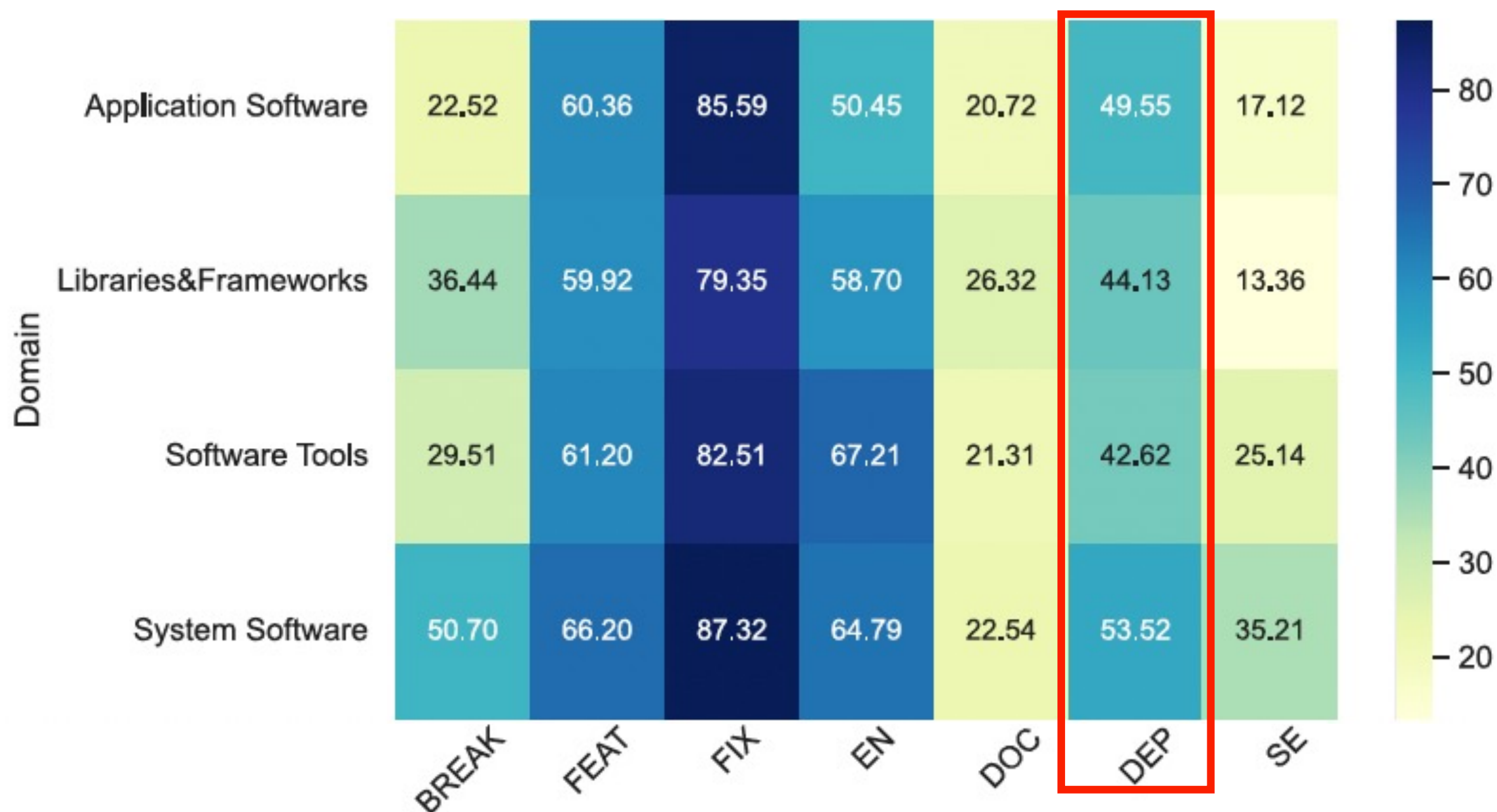
# 調査結果 | 文体

説明文, 記述文, 説得文の3種類があり, メジャーリリースの59.26%は, 明確さと読みやすさを向上させるために, 説得文を利用している

| Level                    | Description                                                                                                                                                                                                                                                   | Major. | Minor. | Patch. | Total. |
|--------------------------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|--------|--------|--------|
| Expository <sup>a</sup>  | RN producers directly list the content (usually title) of change-related commits/PRs/issues                                                                                                                                                                   | 14.20% | 29.96% | 41.70% | 30.07% |
| Descriptive <sup>b</sup> | RN producers re-phrase the content of change-related commits/PRs/issues to increase the readability. Sometimes RN producers summarize the content of similar commits/PRs/issues.                                                                              | 26.54% | 33.48% | 42.15% | 34.80% |
| Persuasive <sup>c</sup>  | In addition to presenting the content of related commits/PRs/issues, RN producers provide additional information to help developers understand the changes, such as the rationale behind the changes, the impact of the changes, and guides for the upgrades. | 59.26% | 36.56% | 16.14% | 35.13% |

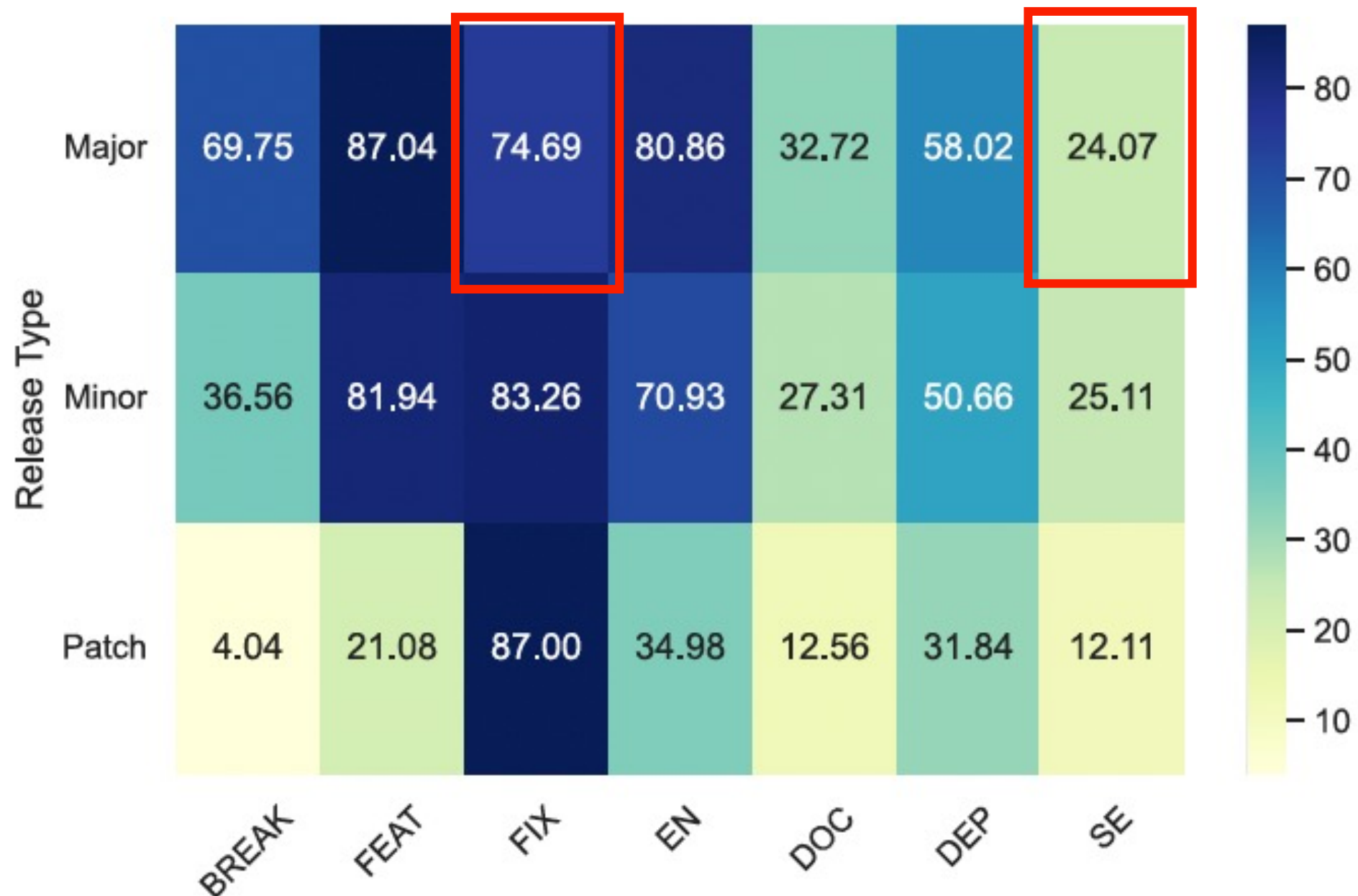
# 調査結果 | 内容

依存関係/環境の変化も半数が文書化されており、インストールの失敗を避けるためだと考えられる



# 調査結果 | 内容

マイナーと比べてメジャーでは、修正されたバグとセキュリティの変更は少ない



# 考察 | 構造

| 対象によって構造を考慮する必要がある

- エンドユーザ：変更タイプと変更優先度
- 開発者：影響を受けるモジュール

| ハイブリッド戦略を用いることも有効であるが、  
以下の問題に注意する必要がある

- かかる労力がより大きくなる
- 複数の構造を利用すると、情報が分散し特定の種類の情報を求めるユーザを混乱させる恐れがある

# 考察 | 文体

変更点の数によって文体を考慮する必要がある

- 変更点が少ない場合
  - 説明文スタイルを推奨
- 変更点が多い場合
  - 記述文スタイルや、説得文スタイルを推奨
  - ただし、これらのRNを自動生成するツールは現在なし

また、丁寧に書かれたRNを読むと、ユーザは誠実さを感じ、更新意欲が高まることが明らかになっている[7][8]

[7] S. L. Abebe, N. Ali, and A. E. Hassan, “An empirical study of software release notes,” *Empirical Software Engineering*, vol. 21, no. 3, pp. 1107–1142, 2016.

[8] E. Aghajani, C. Nagy, M. Linares-Vásquez, L. Moreno, G. Bavota, M. Lanza, and D. C. Shepherd, “Software documentation: the practitioners’ perspective,” in *2020 IEEE/ACM 42nd International Conference on Software Engineering (ICSE)*. IEEE, 2020, pp. 590–601.

# 考察 | 内容

- | ライブラリ & フレームワークは、下流プロジェクトの利用を促進するために、ドキュメントの変更にもっと注意を払う必要がある
- | ソフトウェアツールは、開発者のために、パフォーマンスとセキュリティの改善にもっと焦点を当てるべきである
- | また、ある開発者は、「バグ修正（パッチ）と新機能（マイナー）および破壊的変更（メジャー）は、別々のリリースに分けるのが望ましい」と述べている[9]

[9] “Reason for minor vs patch rules in semver,” <https://stackoverflow.com/questions/46720398/reason-for-minor-vs-patch-rules-in-semver>.

# 考察 | リリースノートの自動生成

| 構造については、完全に自動化することはできない

- 厳格なコミットルールや、PR/issueラベリングシステムが必要
- 変更をさまざまなタイプの内容に自動的に分類し、影響を受けるモジュールにリンクするモデルが必要

| 文体については、半自動的生成が試みられている

- しかし、実際の開発環境で適用されたことはない
- 変更の根拠や目標、および変更の効果の生成
  - コミット、PR、issue、マイルストーンなどのマイニング
  - キー情報抽出、テキスト要約などのNLP技術

# 結論

- | 233の人気GitHubソフトウェアプロジェクトから612の最新RNを手動で分析し，構造，文体，内容の次元からRNを特徴付けた
- | 変更をそれぞれ以下の3つに特定した
  - 構造
    - 変更タイプ，影響を受けるモジュール，変更の優先順位
  - 文体
    - 説明的，記述的，説得的