

Feature Toggles: Practitioner Practices and a Case Study

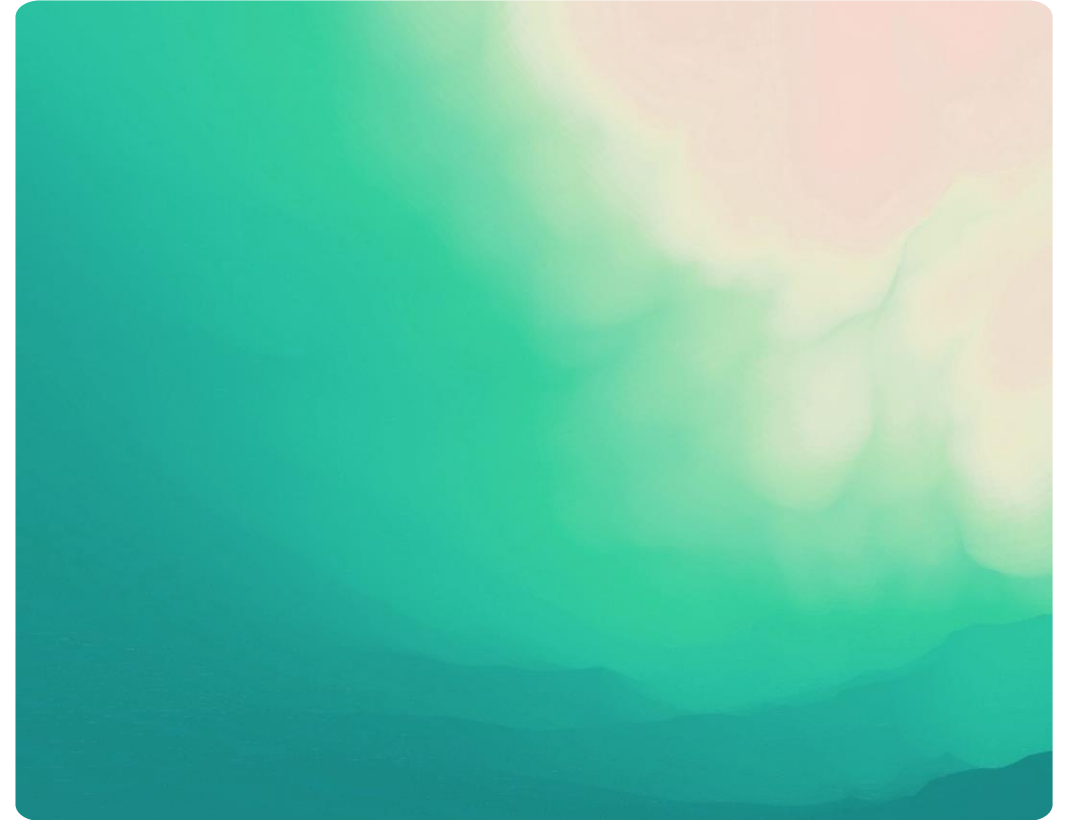
Md Tajmilur Rahman

Louis-Philippe Querel

Peter C. Rigby


Bram Adams

Concordia University, Polytechnique Montreal





THE PROBLEMATIC



How to make integration faster?

- Continuous delivery & Rapid releases
- Reduce probability of integration conflicts
- **Feature toggles**
- 6 weeks (Chrome)
- 2 weeks (Facebook Mobile)
- Daily (Netflix, Facebook Web)

What is Feature Toggles?

```
compositor_switches.cc (chrome/ui/compositor)
content_switches.cc (chrome/content/public/common)
gaia_switches.cc (chrome/google_apis/gaia)
```

(a)

```
107 // Disable 3D inside of flapper.
108 const char kDisableFlash3d[] = "disable-flash-3d";
109
110 // Disable using 3D to present fullscreen flash
111 const char kDisableFlashFullscreen3d[] = "disable-flash-fullscreen-3d";
```

(b)

Toggles are strings that can be set or unset

```
452 if (command_line->HasSwitch(switches::kDisableFlashFullscreen3d))
453     return;
```

(c)

Switch files instead of single configuration file

- 1) A configuration file with all feature toggles and their state (value).
- 2) A mechanism to switch the state of the toggles effectively enabling or disabling a feature.
- 3) Conditional blocks that guard all entry points to a feature such as changing a toggle's value can enable or disable the feature's code.

There are also risks...

- **Dead code** ("comments out" large blocks of code)
- **Combinatorial explosion** of possible sets that need to be tested





THE PROBLEMATIC



GOAL OF THE STUDY

Our goal

- No empirical study that examines how it's used.
- Better understanding of feature toggles (benefits & risks)





THE PROBLEMATIC



GOAL OF THE STUDY



METHODOLOGY

Why did they choose Google Chrome?

- 2.4 thousands distinct toggles
- Most developers on Chrome are based in Montreal (check findings)



The first stage (Quantitative)

- 1) Adoption: How many toggles exist?
- 2) Usage: How are toggles used?
- 3) Development Stage: Are toggles modified during development or release stabilization?
- 4) Lifetime: How long do toggles remain in the system?



The second stage (Quantitative)

- Toggle maintenance & Toggle Debt
- Studying the impact of a toggle maintenance campaign



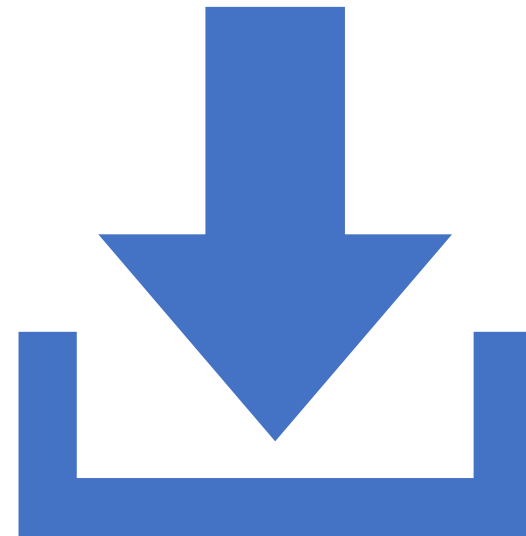
The third stage (Qualitative)

- Examining talks & writing of prominent release engineers (Twitter, Facebook, Google)



How did they find the data? (Quantitative)

- Mining Chrome's Version History
 - 2010 – 2015
 - 39 releases from release 5 to 43
 - Main data source: Git Version control system
- Mining Toggle Maintenance Spreadsheet of Chrome
 - Toggle Maintenance on release 35
 - Study the technical debt & classification



How did they find the data? (Qualitative)

- Talks of 16 prominent release engineers at 13 companies
 - Examined talks & papers from 3 editions of RELENG
 - Online searches with “feature toggles”, “feature flags”, “feature switches” and “feature flippers”
 - 17 talks and blog posts from 13 big companies (Google, Netflix)
1. First, we printed blog posts and paraphrased talks. For the talks, we included timing information to have a direct link to the original evidence.
 2. To code the artifacts, we wrote notes in the margins. We cut the printed material to divide it into separate groups of emerging codes. We then progressively repeated this grouping and comparison step, attaining more abstract codes that we recorded on memo pads.
 3. We continued to sort, compare, group, and abstract codes until we obtained a set of high-level themes that can each be traced back to practitioner statements, and explain how practitioners perceive the use of feature toggles.

How to determine the validity?

- Member checking with 4 Google Chrome developers





THE
PROBLEMATIC



GOAL OF THE
STUDY



METHODOLOGY



RESULTS

Adoption: How many toggles exist? (First stage)

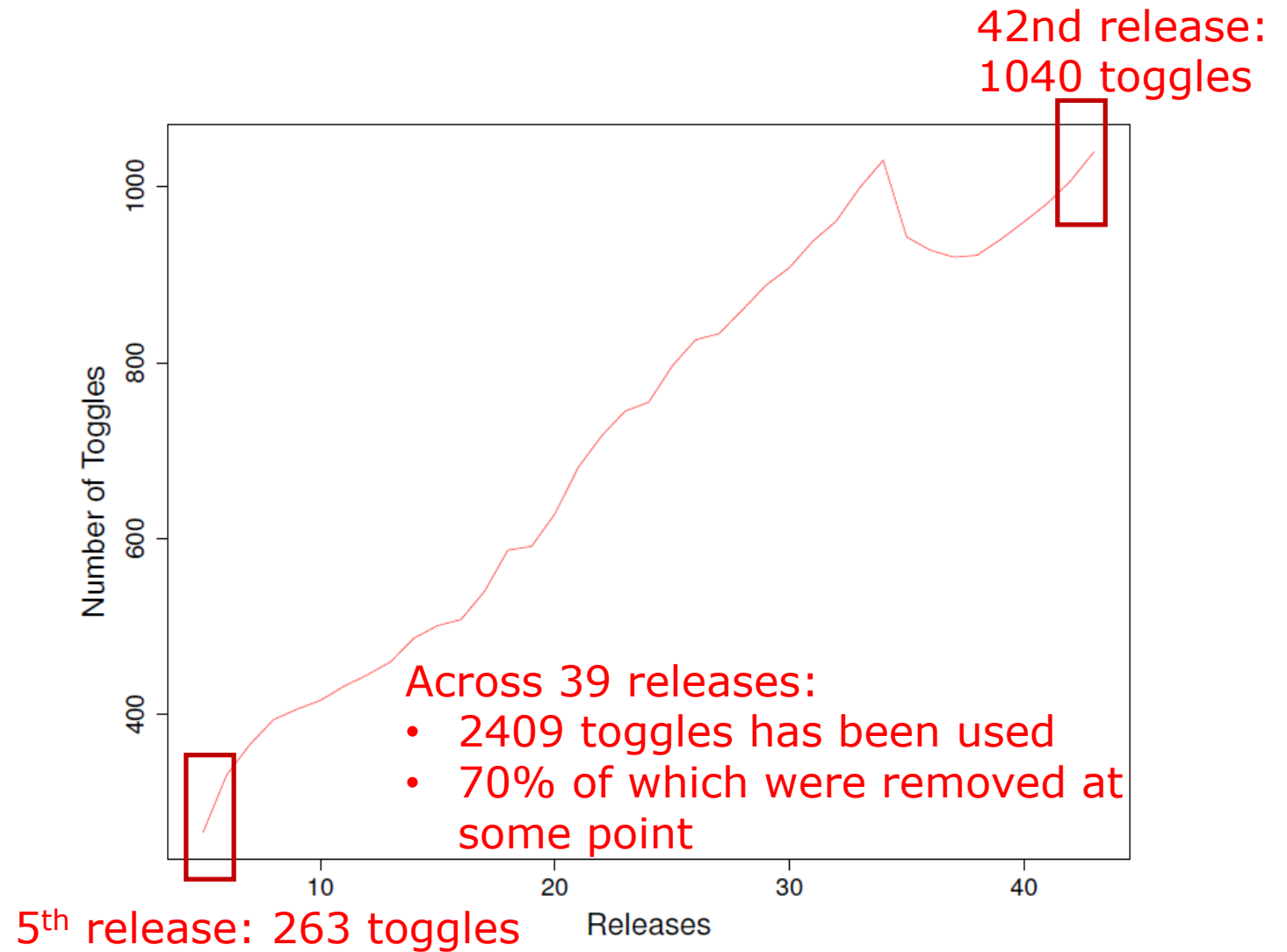


Figure 2: Number of unique toggles per release of Google Chrome.

- Growing with a median increase of 30 toggles per release
- Each release sees a median of 73 added and 43 removed.

Adoption: How many toggles exist? (First stage)

"The number of toggles increases linearly over time, except for a period of active maintenance at release 35."

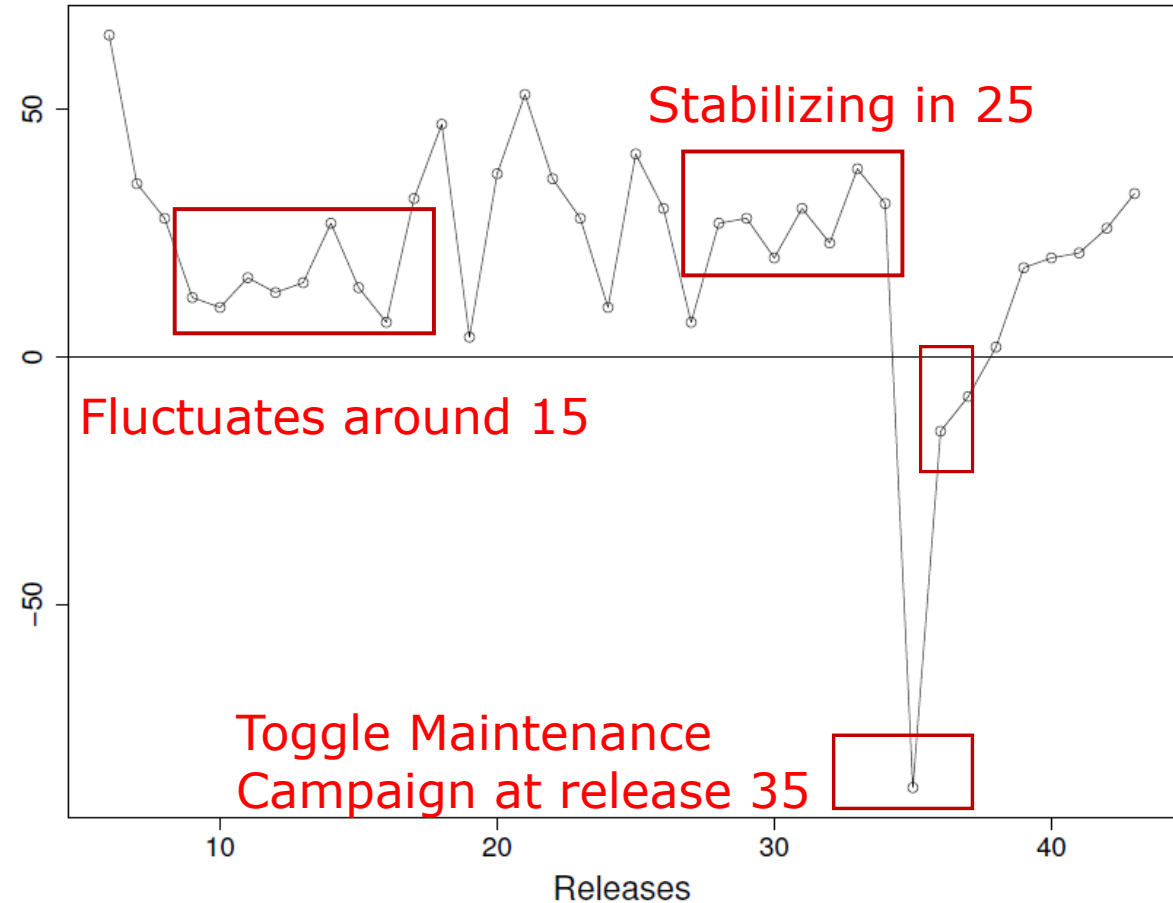


Figure 3: Change in number of unique toggles across releases of Google Chrome. A negative value means that a release has less toggles than the previous one.

Usage: How are toggles used? (First stage)

- Directly used in a conditional statement
- Assigned to a variable that is used later

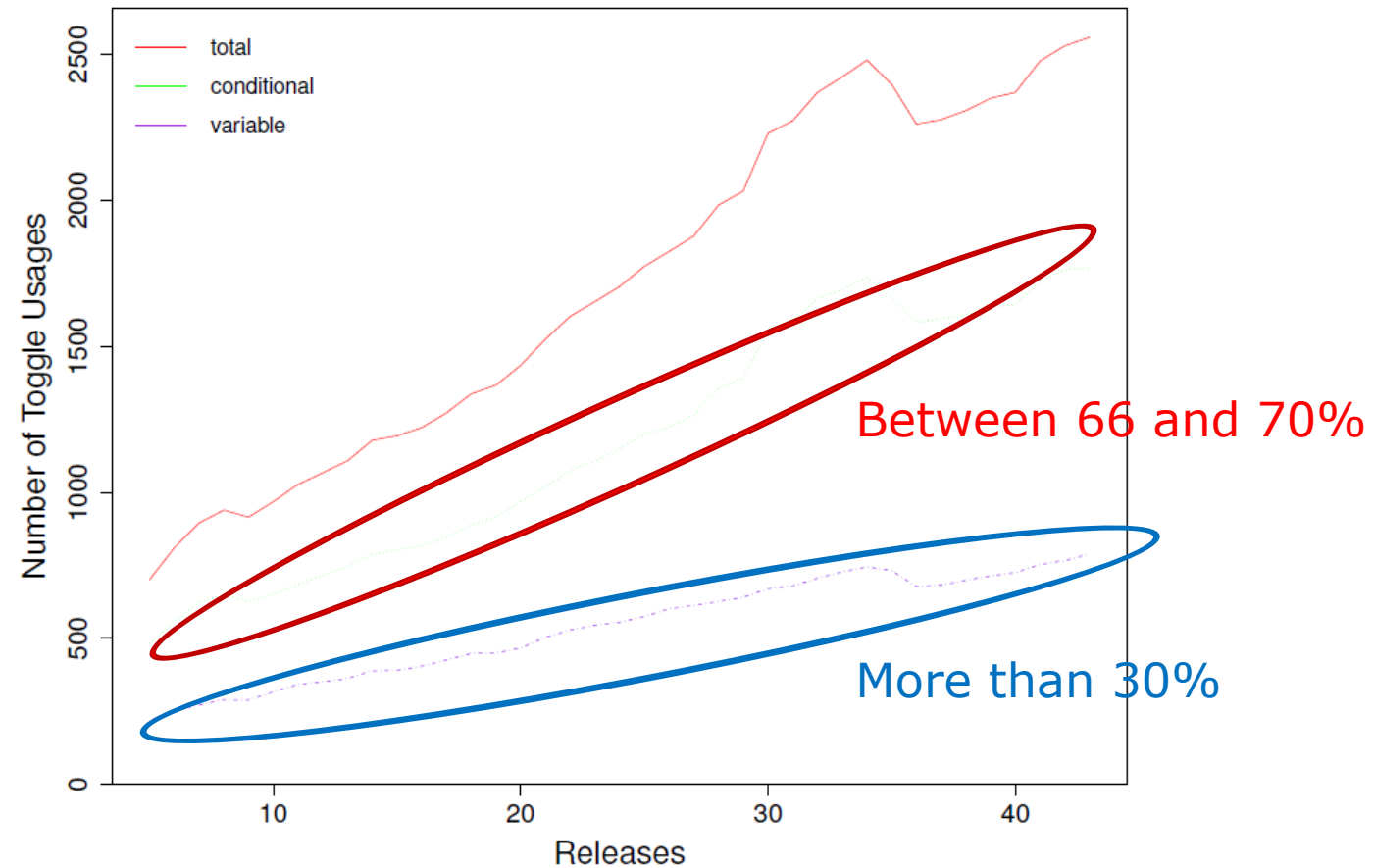


Figure 4: The total number of times toggles are used per release of Google Chrome as well as the type of usage: direct conditional or variable assignment.

What are toggles assigned to a variable?

- Allows more flexible and complex usages

Repeatedly used to adjust the behaviour of the system



```
99  return command_line.HasSwitch(switches::kInProcessPlugins) ||  
100  command_line.HasSwitch(switches::kSingleProcess);
```

```
453  base::CommandLine* cmdline = base::CommandLine::ForCurrentProcess();  
454  config.disable_auto_update =  
455  cmdline->HasSwitch(switches::kSbDisableAutoUpdate) ||  
456  cmdline->HasSwitch(switches::kDisableBackgroundNetworking);  
457  config.url_prefix = kSbDefaultURLPrefix;  
458  config.backup_connect_error_url_prefix = kSbBackupConnectErrorURLPrefix;  
459  config.backup_http_error_url_prefix = kSbBackupHttpErrorURLPrefix;  
460  config.backup_network_error_url_prefix = kSbBackupNetworkErrorURLPrefix;
```

Usage: How are toggles used? (First stage)

"Most toggles are used directly in a conditional, however, at least 30% of toggles are assigned to a variable to allow them to significantly change the behaviour of Chrome."

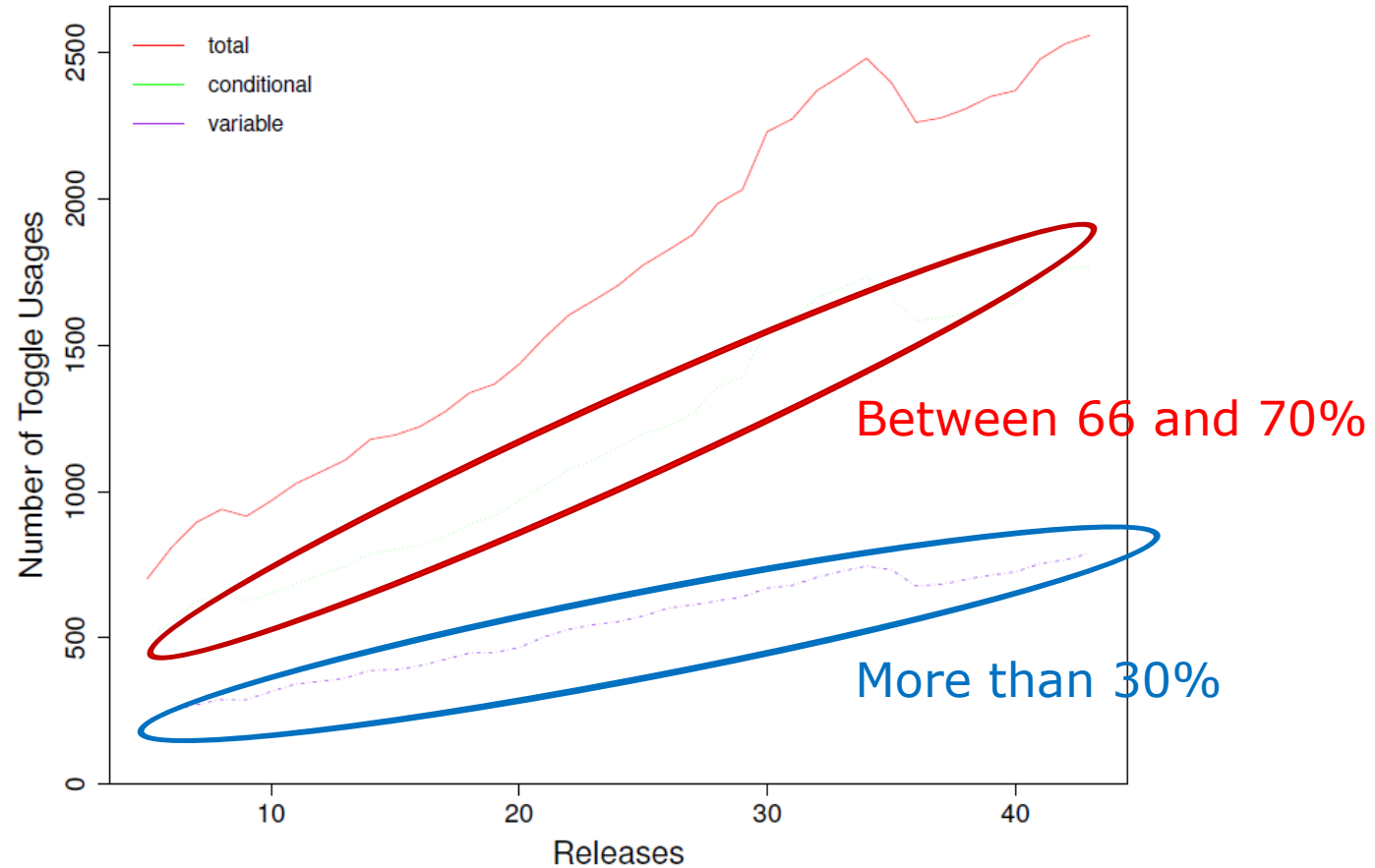



Figure 4: The total number of times toggles are used per release of Google Chrome as well as the type of usage: direct conditional or variable assignment.



Development Stage: Are toggles modified during development or release stabilization? (First stage)

- Google releases new version of Chrome every 6 weeks
 - Feature freeze (small group of maintainers)
 - Feature not ready → stabilization channel
- From 5000 commits
 - 97% of the toggling events occur during development
 - 3% occur during release stabilization

"Most toggle changes occur during development, only 3% of all toggle changes happen during release stabilization."

Lifetime: How long do toggles remain in the system? (First stage)

- On average 72% of the toggles survive 5 or more releases.
- Some toggles are intended for long-term use
 - Toggleable business function
 - Intended to be removed after feature has stabilized
 - Introduced for a period of less than a release to isolate wip

"In general, the lifetime for toggles is long, with half of the toggles surviving 12 or more releases"

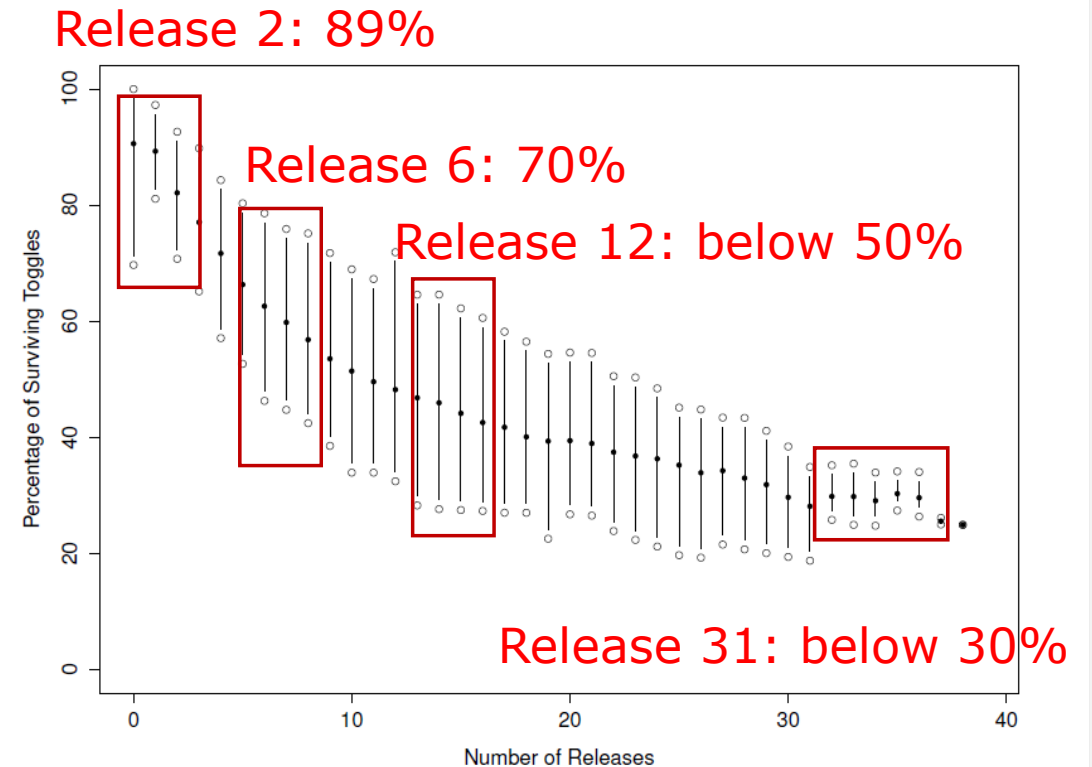
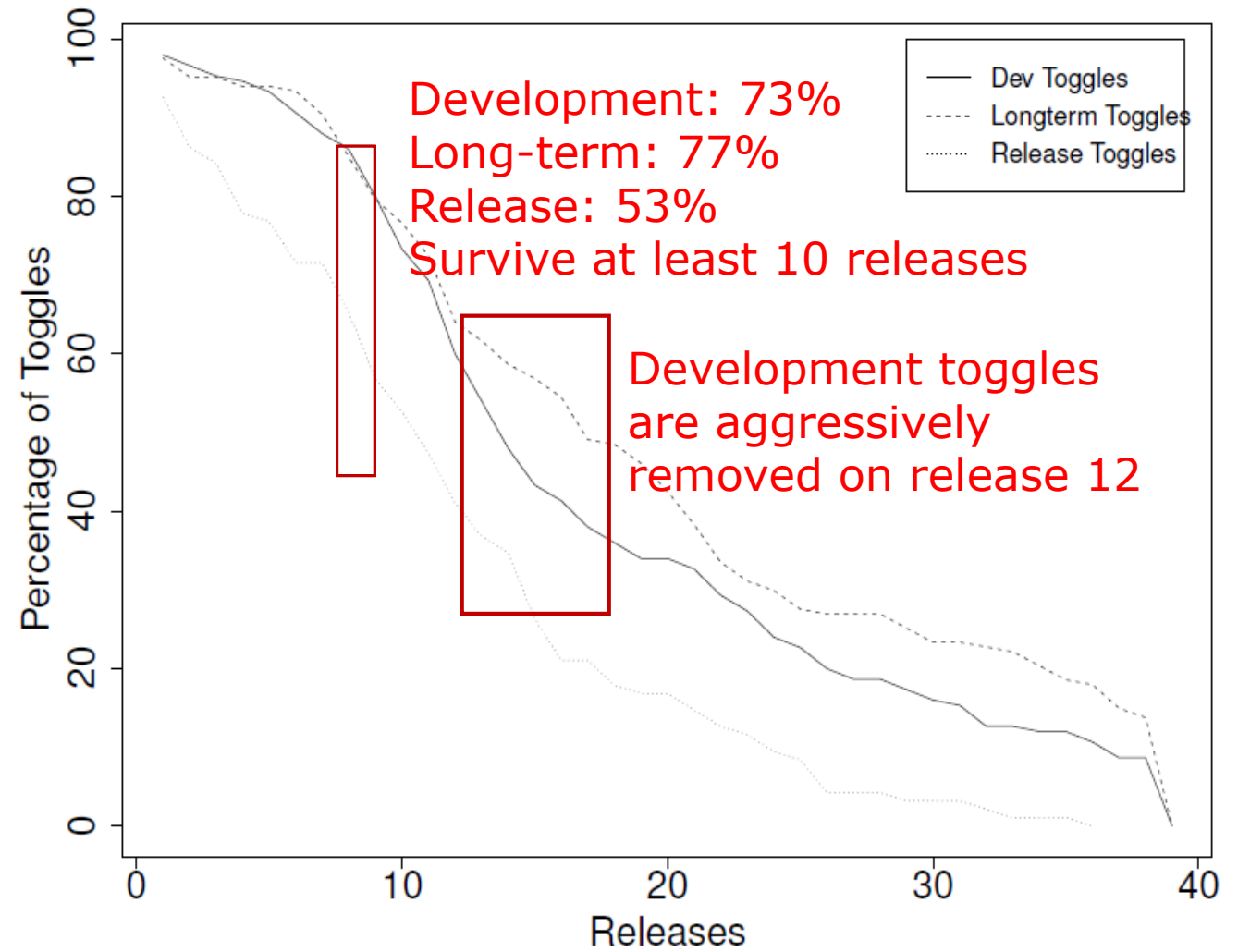


Figure 7: Survival curve showing the percentage of toggle that remain in the code base for 1, 2, ..., 38 releases of Google Chrome. Each vertical bar spans between the 5th and 95th percentile, with the average as the middle point.

What happened during Toggle Maintenance Campaign? (Second stage)

- **Development Toggles:** Testing & Debugging
- **Long-term business Toggles:** Toggles different features to different users
- **Release Toggles:** Gradual roll-out of new features to ensure a feature is ready for production
- **Weakness:**
 - Toggles should be removed after a feature has been stabilized (lingering existence is toggle debt)
 - Hard to convince developers to remove them
 - Lack of tool support for identifying all if-conditions
 - Developers have moved on new features while previous feature takes a few releases before considering stable



"There are three types of toggles: development, long-term business, and release toggles. Although release toggles are shorter lived than the other types of toggles, 53% still exist after 10 releases indicating that many linger as technical debt."

What do the experts think? (Third stage)

Reconciling Rapid Release and Long-term Feature Development

- Google (6 weeks), Facebook (twice/day)
- Limit scope of feature & requires teams to continuously integrate
- Big features takes months and clashes against rapid release strategy.
- Final merge can take unpredictable amount of time
- **Solutions: Feature Toggle**

Flexible Feature Roll-out

- Flexibility to gradually roll out features and do user “experiments” (A/B testing)
- Facilitate gradual roll out new features (canary deployment)

What do the experts think? (Third stage)

Enabling Fast Context Switches

- Broken builds are caused by committing to wrong branches (Kerzazi et al.)
- Features toggles allow developers to prioritize on feature and bug they want to work on & Toggling requires less effort than switching branches (Ho)

Features are Designed to be Toggleable

- Requires a shift in mentality (feature to be easy to revert)
- Features need to be isolated to avoid toggle dependencies
- Cross-cut large number of components (less modular)
- Feature independent & decoupled from other features

What does the experts think? (Third stage)

Toggle Debt

- Dead code
- Preliminary code (Code behind toggles have lesser quality)

Combinatorial Feature Testing

- Big companies have massive test suite on changes made to master
- Explosion of tests to run
- “How do we test all possible combinations of features?” – Practitioners
- Only test combinations that one realistically is going to use

Conclusion

- Feature toggles: Enable or disable new features (wip) → simplify merging
- 3 major contributions:
 - Quantify the prevalence and lifecycle of toggles
 - Categorize types of toggles, Technical Debt of Toggle maintenance campaign
 - How practitioners use toggles
 - **Reconciling Rapide Release and Long-term Feature Development**
 - **Flexible Feature Roll-out**
 - **Enabling Fast Context Switches**
 - **Designing Ffeatures to be Toggleable**
 - **Toggle Debt**
 - **Combinatorial Feature Testing**