



Do Energy-oriented Changes Hinder Maintainability?

Luís Cruz, Rui Abreu, John Grundy, Li Li, Xin Xia



MONASH
University

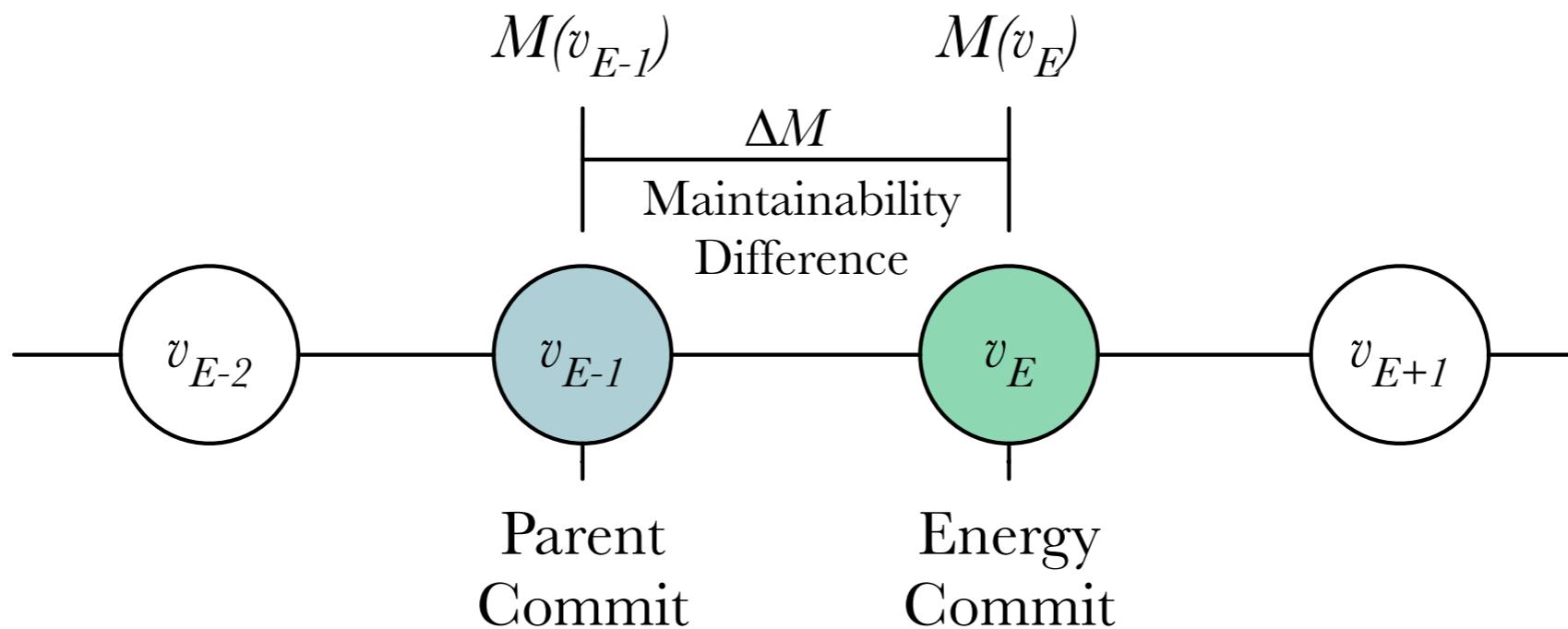


Motivation

- If users lose connectivity they might be unable to accomplish **important tasks**
- Users tend to **remove apps** that drain the phone battery
- Developers need to **build energy-efficient** mobile apps
- Improving energy-efficiency is **not trivial**: requires specialised knowledge and implementing code changes

Maintainability of Energy Changes

- What is the impact of making energy-oriented code changes on the maintainability of mobile apps?



Measuring Maintainability

- According to ISO/IEC 25010, Maintainability is “*the degree of effectiveness and efficiency with which a software product or system can be modified to improve it, correct it or adapt it to changes in environment, and in requirements*”
- We use the code analysis tool Better Code Hub to assess maintainability
- Better Code Hub maps the ISO/IEC 25010 standard on maintainability into a set of guidelines derived from static analysis

The screenshot shows a web-based code analysis tool named NetGuard. At the top, there's a circular badge indicating 'Compliance 5 of 10'. The repository information is shown as 'luiscruz/NetGuard' with a note 'Last analysis: 3 months ago' and 'Branch: energy_test_fd614bc (default)'. A gear icon for settings is also present.

The main content area displays a 'GUIDELINE EXPLANATION' for the rule 'Write Short Units of Code'. It lists 'Refactoring candidates' under the 'Unit' category:

	Lines of Code
<input type="checkbox"/> AdapterRule.onBindViewHolder(ViewHolder, int)	280
<input type="checkbox"/> ActivityMain.onCreate(Bundle)	204
<input type="checkbox"/> ActivitySettings.onPostCreate(Bundle)	202
<input type="checkbox"/> StatsHandler.updateStats()	174
<input type="checkbox"/> ActivitySettings.onSharedPreferenceChanged(SharedPreferences, ...)	164
<input type="checkbox"/> AdapterLog.bindView(View, Context, Cursor)	151
<input type="checkbox"/> Rule.getRules(boolean, Context)	151
<input type="checkbox"/> ActivityLog.onCreate(Bundle)	149
<input type="checkbox"/> DatabaseHelper.onConfigure(SQLiteDatabase, int, int)	115

A horizontal bar at the bottom indicates the distribution of lines of code: a green segment (0-15), a yellow segment (16-30), an orange segment (31-60), and a red segment (more than 60). The red segment is the longest, representing the majority of the listed methods.

Below this section, there's a list of other guidelines with their status (green checkmark or red X):

- Write Simple Units of Code: X
- Write Code Once: ✓
- Keep Unit Interfaces Small: X
- Separate Concerns in Modules: X
- Couple Architecture Components Loosely: ✓
- Keep Architecture Components Balanced: ✓
- Keep Your Codebase Small: ✓

**luiscruz/
NetGuard**

Last analysis: 3 months ago Branch: energy_test_fd614bc (default)

Compliance 5 of 10

Write Short Units of Code

GUIDELINE EXPLANATION

Refactoring candidates

Unit

	Lines of Code
AdapterRule.onBindViewHolder(ViewHolder, int)	280
ActivityMain.onCreate(Bundle)	24
ActivitySettings.onPostCreate(Bundle)	202
StatsHandler.updateStats()	17
ActivitySettings.onSharedPreferenceChanged(SharedPreferences, String, String)	164
AdapterLog.bindView(View, Context, Cursor)	151
Rule.getRules(boolean, Context)	151
ActivityLog.onCreate(Bundle)	149
Databasell... onL... (SOL...)	115

Threshold Marks

at most 15 lines of code more than 15 lines of code more than 30 lines of code more than 60 lines of code

Write Simple Units of Code

Write Code Once

Keep Unit Interfaces Small

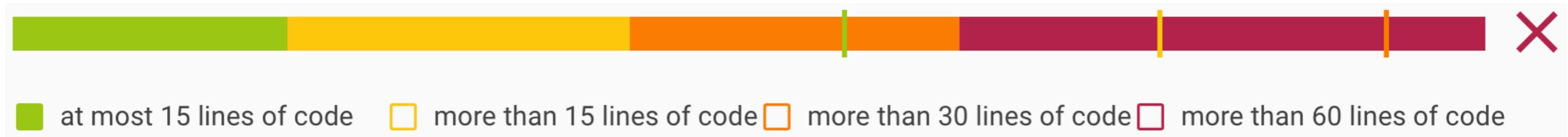
Separate Concerns in Modules

Couple Architecture Components Loosely

Keep Architecture Components Balanced

Keep Your Codebase Small

Numerical Score of Maintainability for a Software Version

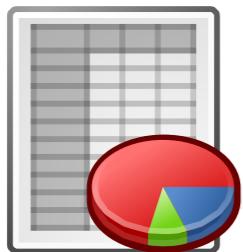


$$\begin{aligned} \text{Maintainability} = & \frac{1}{3} \left(SLOC_{green} - \frac{1 - T_{yellow}}{T_{yellow}} (SLOC_{yellow} + SLOC_{orange} + SLOC_{red}) + \right. \\ & + SLOC_{green} + SLOC_{yellow} - \frac{1 - T_{orange}}{T_{orange}} (SLOC_{orange} + SLOC_{red}) + \\ & \left. + SLOC_{green} + SLOC_{yellow} + SLOC_{orange} - \frac{1 - T_{red}}{T_{red}} SLOC_{red} \right) \end{aligned}$$

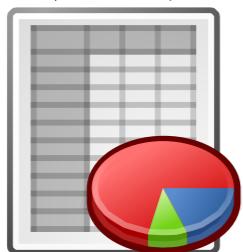
Explained in the paper

Energy Code Changes Dataset

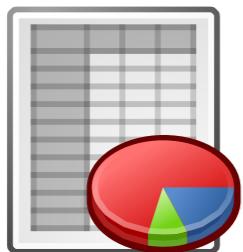
Moura et al.
(2015)



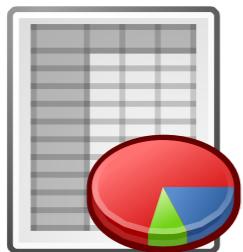
Bao et al.
(2016)



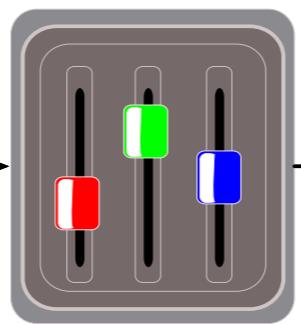
Cruz et al.
(2018)



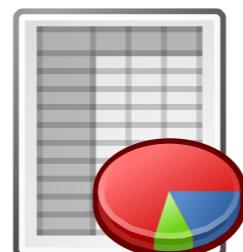
Cruz et al.
(2019)



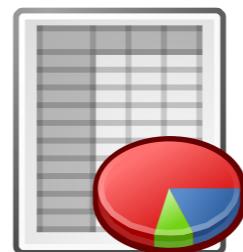
Combine
datasets



Energy
Commits



Baseline
Commits



Better Code Hub
Maintainability

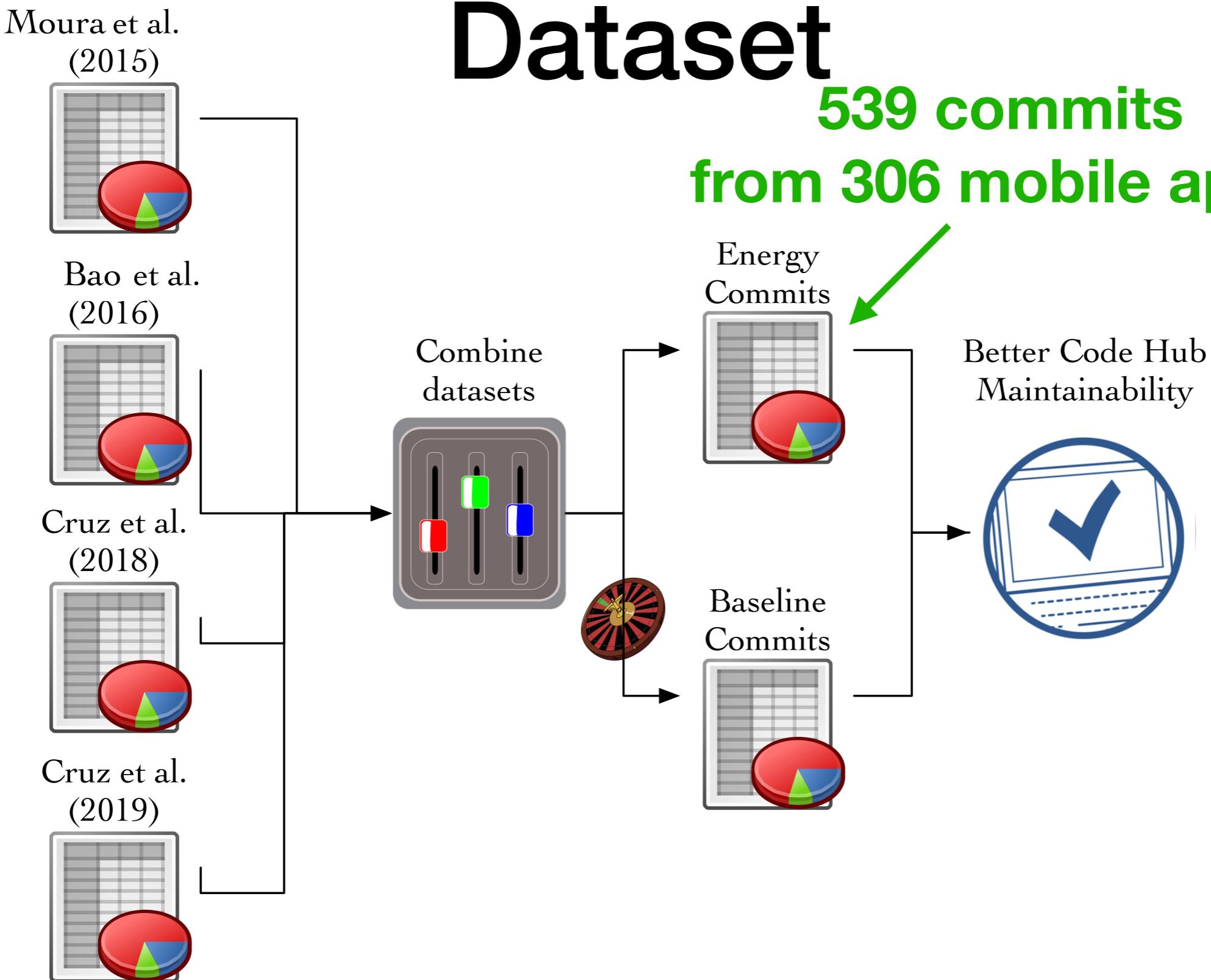


Energy Code Changes

Dataset

539 commits

from 306 mobile apps



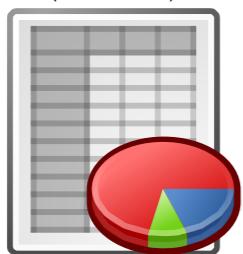
Energy Code Changes

Dataset

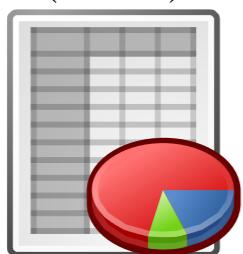
539 commits

from 306 mobile apps

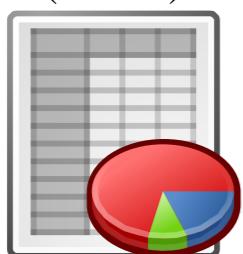
Moura et al.
(2015)



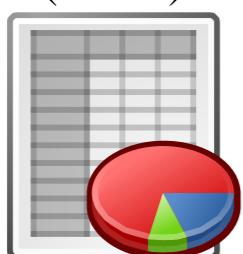
Bao et al.
(2016)



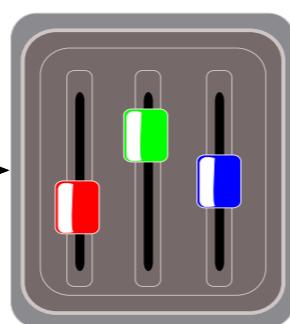
Cruz et al.
(2018)



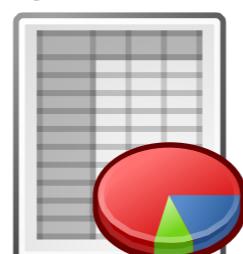
Cruz et al.
(2019)



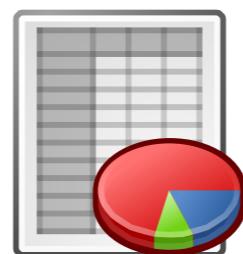
Combine
datasets



Energy
Commits



Baseline
Commits

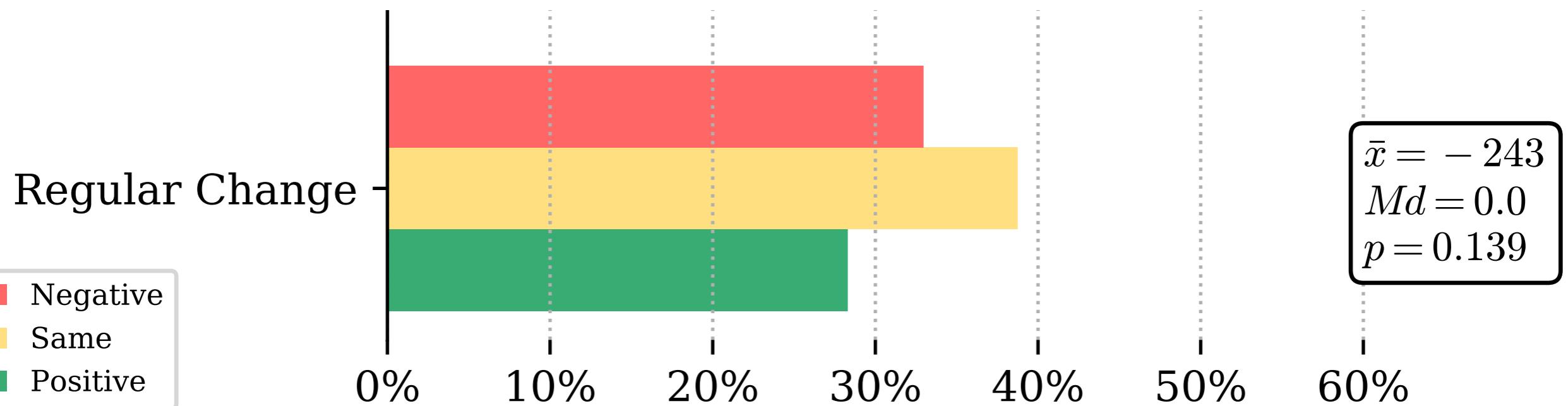


Better Code Hub
Maintainability

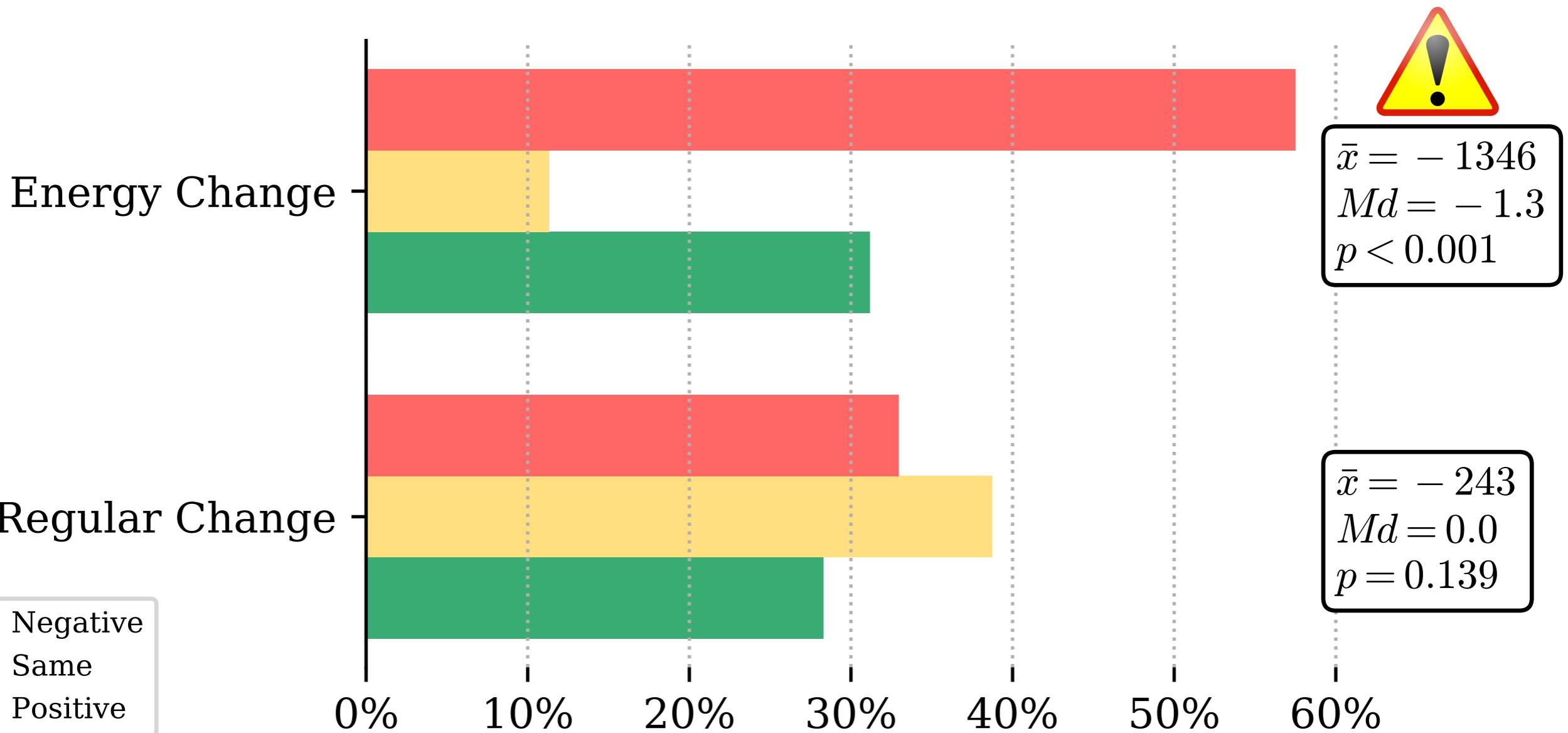


539 baseline commits
from 306 mobile apps

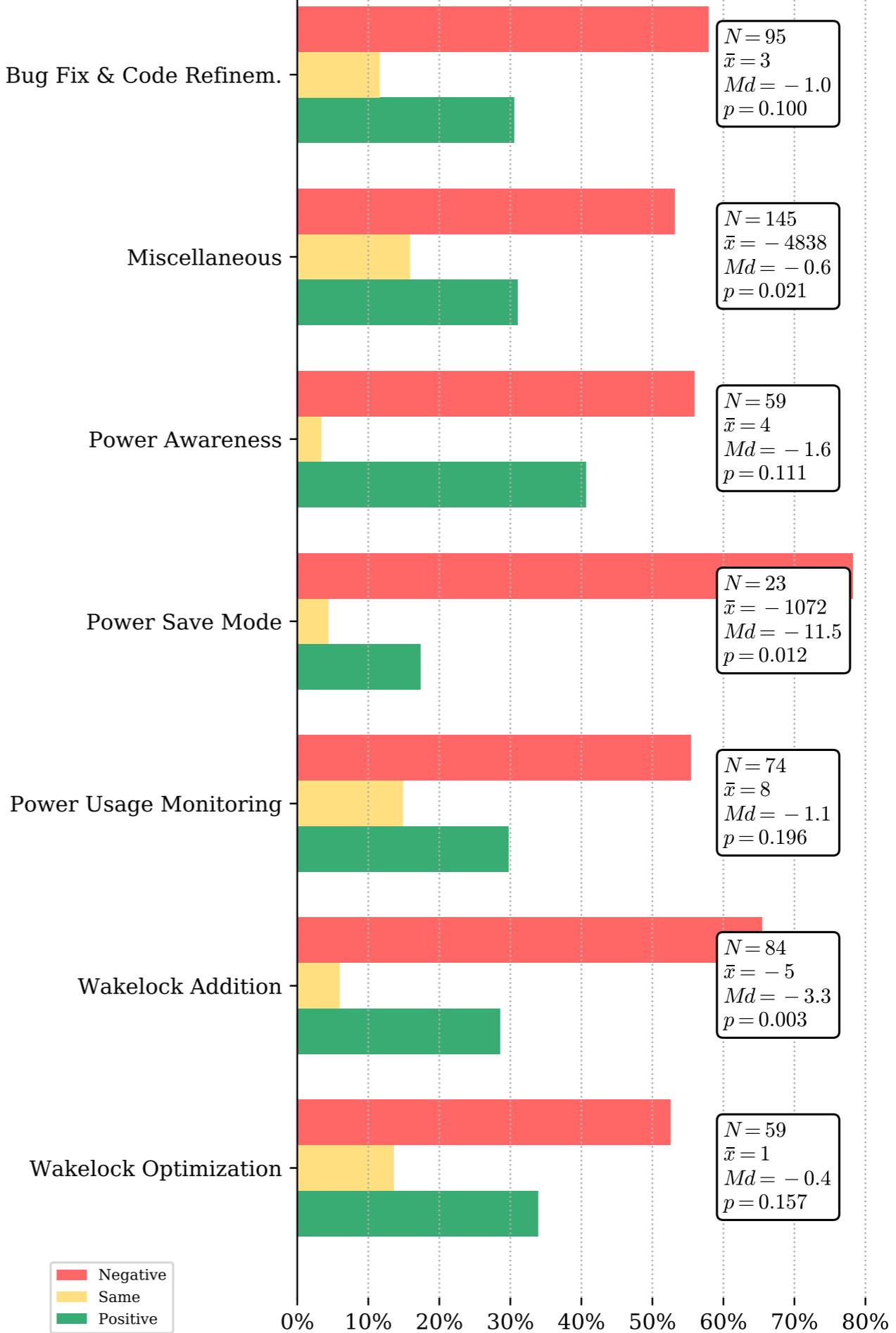
Impact of energy changes on maintainability



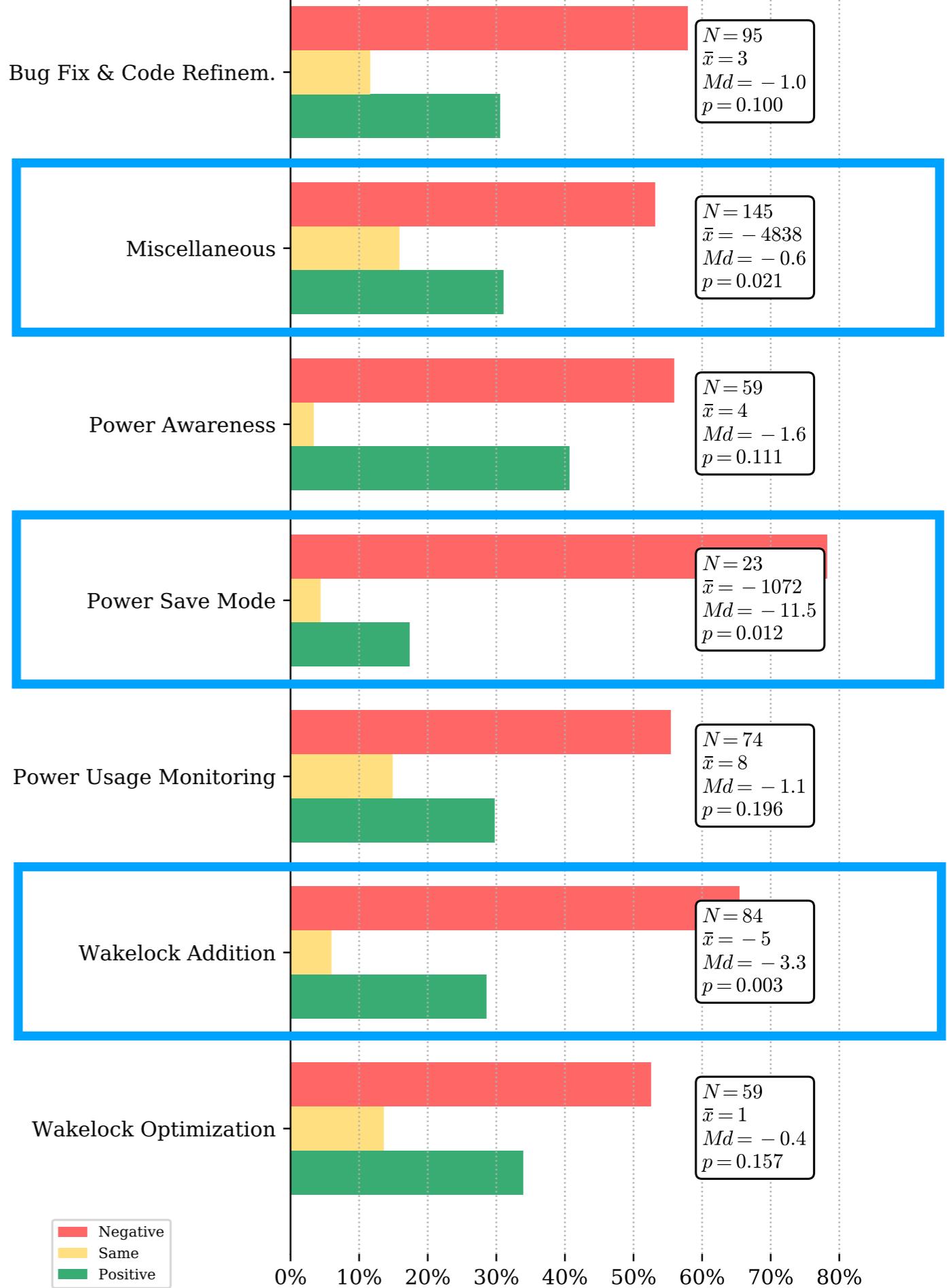
Impact of energy changes on maintainability



Which energy patterns are more likely to affect maintainability?



Which energy patterns are more likely to affect maintainability?



Typical maintainability issue I

<https://github.com/einmalfel/PodListen/commit/2ed5a65>



4 changed files with 28 additions and 0 deletions.

...

```
234     private synchronized void readPreference(Key key) {  
235         switch (key) {  
236             +         case AUTO_DOWNLOAD_AC:  
237                 autoDownloadACOnly = sPrefs.getBoolean(Key.AUTO_DOWNLOAD_AC.toString(), false);  
238                 if (!autoDownloadACOnly) {  
239                     context.sendBroadcast(new Intent(DownloadReceiver.UPDATE_QUEUE_ACTION));  
240                 } else if (!DownloadReceiver.isDeviceCharging()) {  
241                     DownloadReceiver.stopDownloads(null);  
242                 }  
243             +         break;  
244             case PLAYER_FOREGROUND:  
245                 playerForeground = sPrefs.getBoolean(Key.PLAYER_FOREGROUND.toString(), false);  
246             break;  
247         }  
248     }
```

...

Typical maintainability issue II

<https://github.com/mozilla/MozStumbler/commit/6ea0268>



5 changed files with 66 additions and 14 deletions.

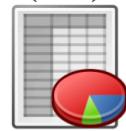
```
161 +
162 +         // each time we reconnect, check to see if we're suppose to be
163 +         // in power saving mode.  if not, start the scanning. TODO: we
164 +         // shouldn't just stopScanning if we find that we are in PSM.
165 +         // Instead, we should see if we were scanning do to an activity
166 +         // recognition.  If we were, don't stop.
167 +         if (mConnectionRemote != null) {
168 +             try {
169 +                 if (mPrefs.getPowerSavingMode()) {
170 +                     mConnectionRemote.stopScanning();
171 +                 } else {
172 +                     mConnectionRemote.startScanning();
173 +                 }
174 +             } catch (RemoteException e) {
175 +                 Log.e(LOGTAG, "", e);
176 +             }
177 +         }
178         updateUI();
179     }
```

Conclusions

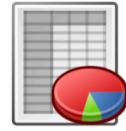
- Energy-oriented commits significantly decrease software maintainability.
- Particularly concerning **Power Save Mode** and **Wakelock Addition patterns**: maintainability decreases in **78%** and **66%** of the cases.
- Mobile development frameworks should provide mechanisms to implement energy patterns

Energy Code Changes Dataset

Bao et al.
(2015)



Moura et al.
(2016)



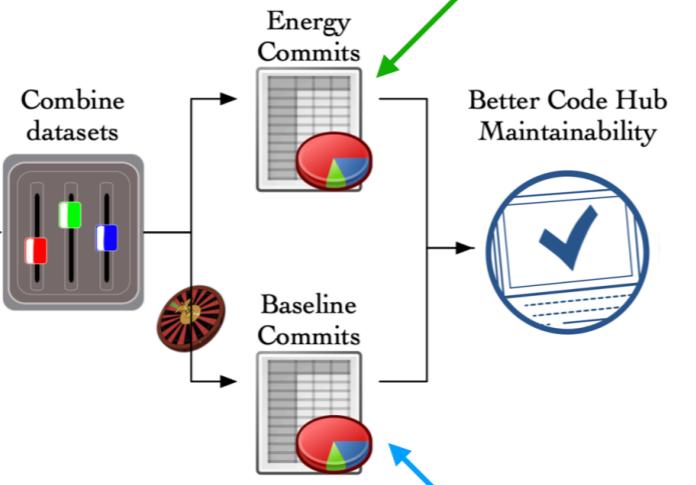
Cruz et al.
(2018)



Cruz et al.
(2019)



539 commits
from 306 mobile apps



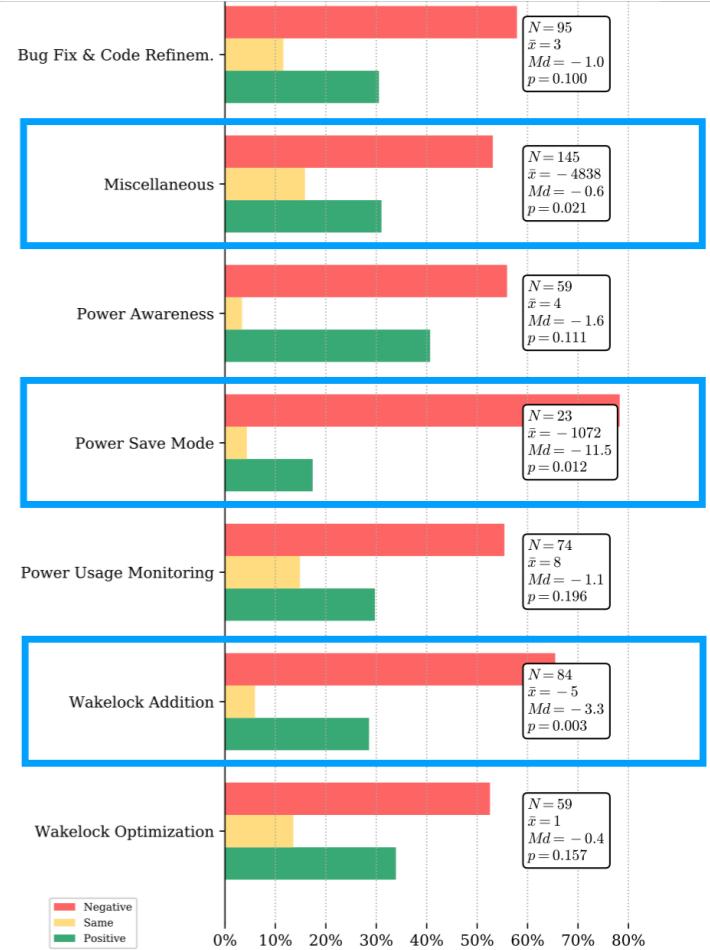
539 baseline commits
from 306 mobile apps

8

Do Energy-oriented Changes Hinder Maintainability?

@luismcruz | ICSME 2019

Which energy patterns are more likely to affect maintainability?



10

@luismcruz | ICSME 2019

Typical maintainability issue I

<https://github.com/einmalfel/PodListen/commit/2ed5a65>

4 changed files with 28 additions and 0 deletions.



```

234
235
236 +     private synchronized void readPreference(Key key) {
237 +         switch (key) {
238 +             case AUTO_DOWNLOAD_AC:
239 +                 autoDownloadACOnly = sPrefs.getBoolean(Key.AUTO_DOWNLOAD_AC.toString(), false);
240 +                 if (!autoDownloadACOnly) {
241 +                     context.sendBroadcast(new Intent(DownloadReceiver.UPDATE_QUEUE_ACTION));
242 +                 } else if (!DownloadReceiver.isDeviceCharging()) {
243 +                     DownloadReceiver.stopDownloads(null);
244 +                 }
245 +                 break;
246 +             case PLAYER_FOREGROUND:
247 +                 playerForeground = sPrefs.getBoolean(Key.PLAYER_FOREGROUND.toString(), false);
248 +                 break;
249 +         }
250 +     }
  
```

@luismcruz | ICSME 2019

