ICSME 2019

Do Energy-oriented Changes Hinder Maintainability?

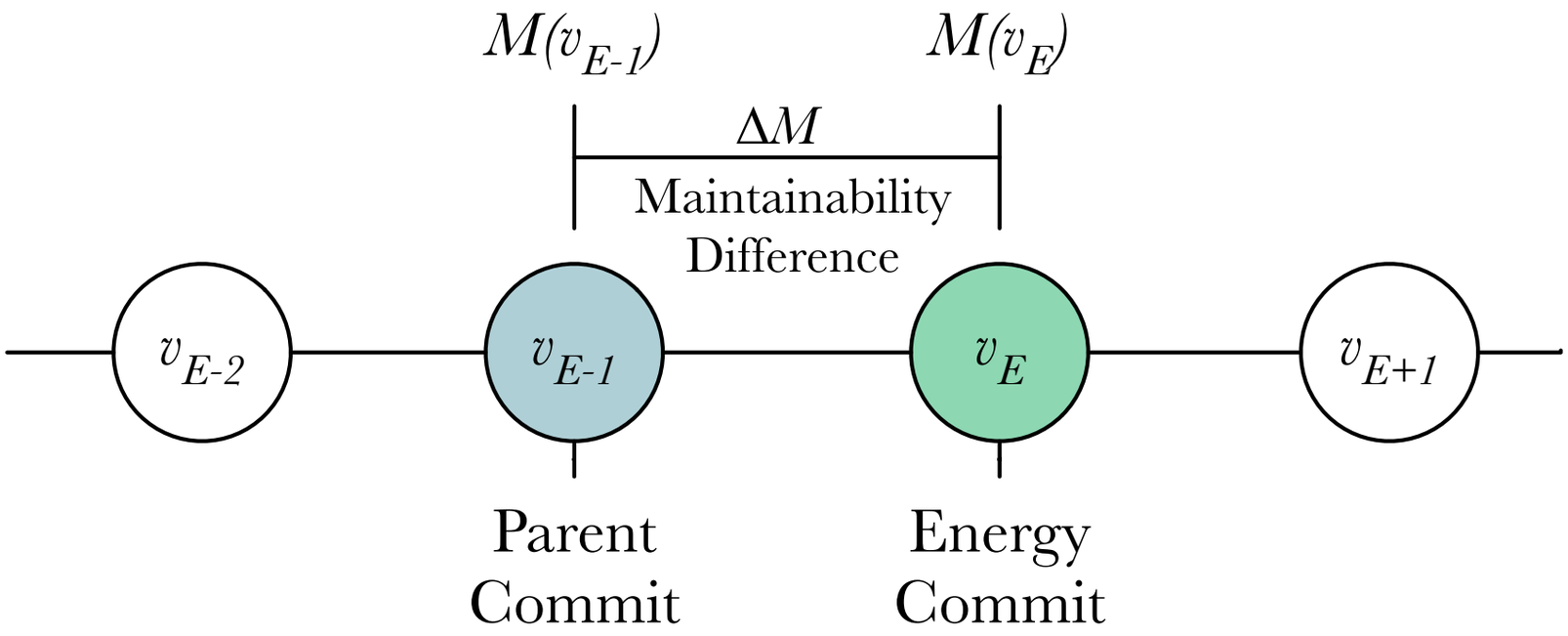
Luís Cruz, Rui Abreu, John Grundy, Li Li, Xin Xia

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?

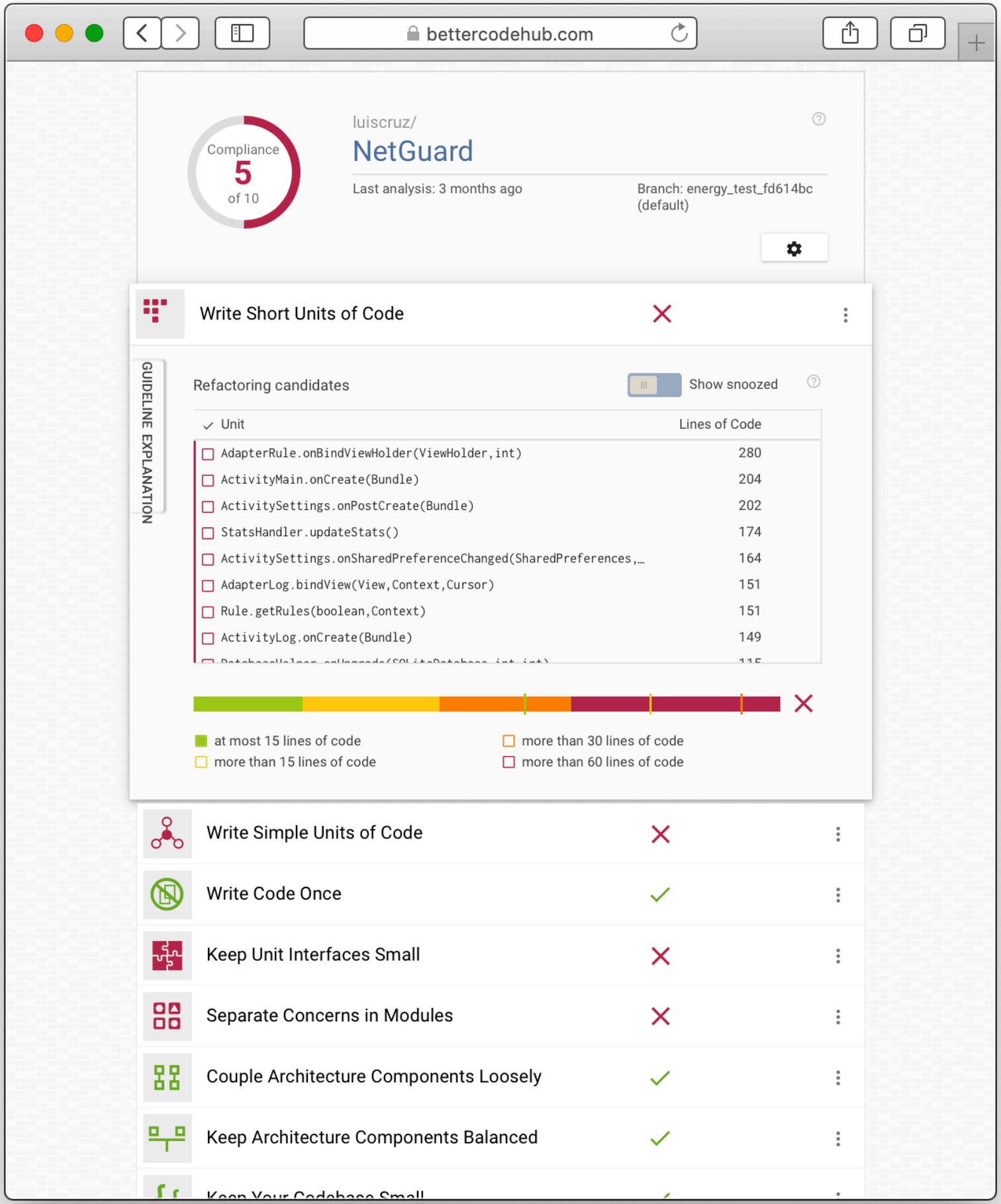


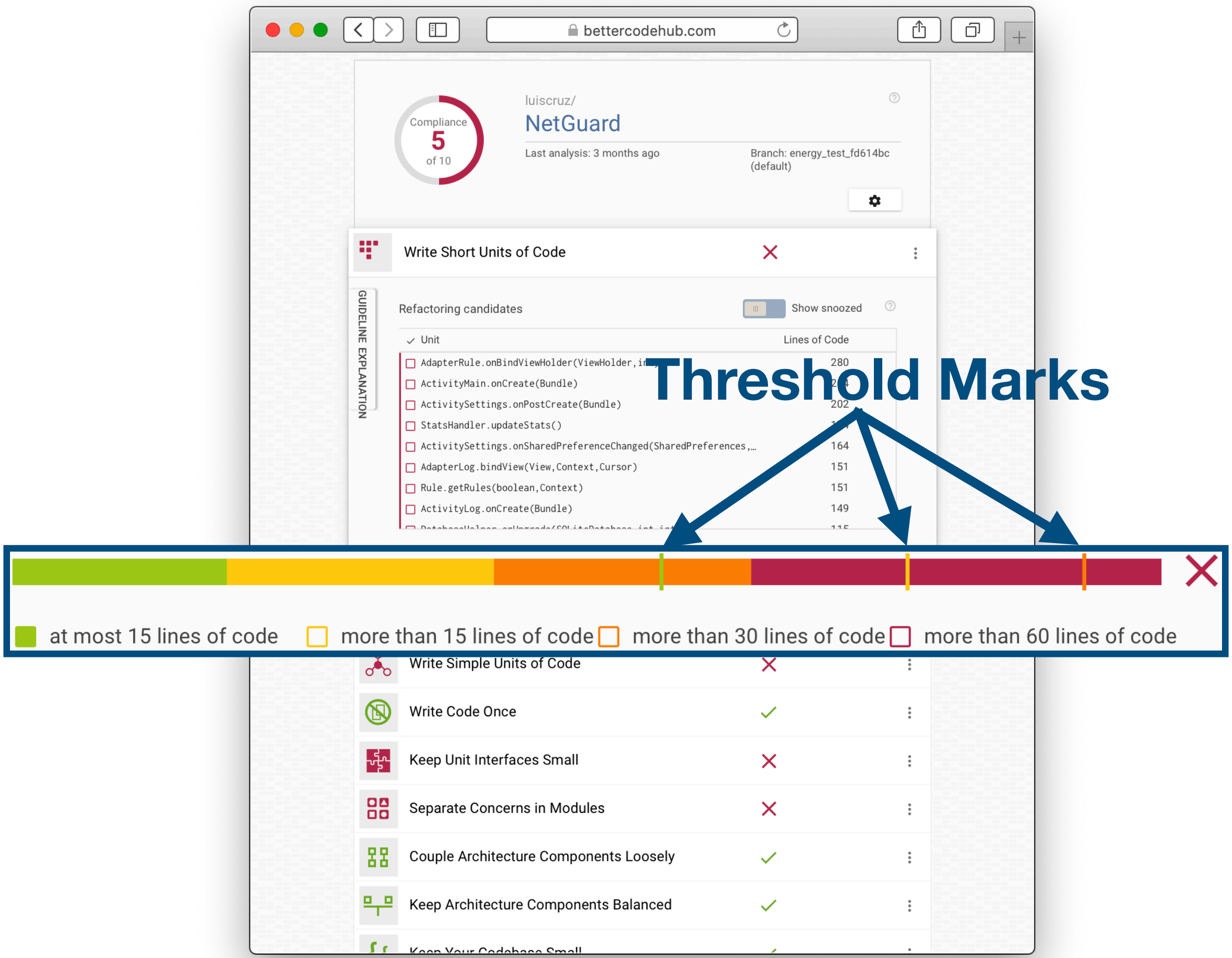
Measuring Maintainability

- 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*"

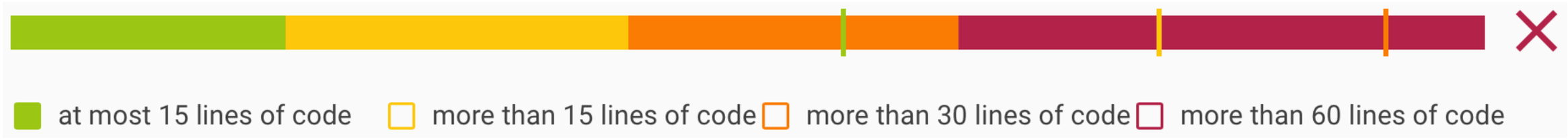
- We use the code analysis tool Better Code Hub to assess maintainability

- Better Code Hub maps the ISO/IEC 25010 standard on maintainability into a set of guidelines derived from static analysis


bettercodehub.com
Compliance
5
of 10
luiscruz/
NetGuard
Last analysis: 3 months ago
Branch: energy_test_fd614bc (default)
Write Short Units of Code
GUIDELINE EXPLANATION
Refactoring candidates
Show snoozed
Unit
Lines of Code
AdapterRule.onBindViewHolder(ViewHolder,int) 280
ActivityMain.onCreate(Bundle) 204
ActivitySettings.onPostCreate(Bundle) 202
StatsHandler.updateStats() 174
ActivitySettings.onSharedPreferenceChanged(SharedPreferences,… 164
AdapterLog.bindView(View,Context,Cursor) 151
Rule.getRules(boolean,Context) 151
ActivityLog.onCreate(Bundle) 149
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


bettercodehub.com
luiscruz/
NetGuard
Compliance
5
of 10
Last analysis: 3 months ago
Branch: energy_test_fd614bc (default)
Write Short Units of Code
GUIDELINE EXPLANATION
Refactoring candidates
Show snoozed
Unit
Lines of Code
AdapterRule.onBindViewHolder(ViewHolder,in
280
ActivityMain.onCreate(Bundle)
ActivitySettings.onPostCreate(Bundle)
202
StatsHandler.updateStats()
ActivitySettings.onSharedPreferenceChanged(SharedPreferences,…
164
AdapterLog.bindView(View,Context,Cursor)
151
Rule.getRules(boolean,Context)
151
ActivityLog.onCreate(Bundle)
149
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

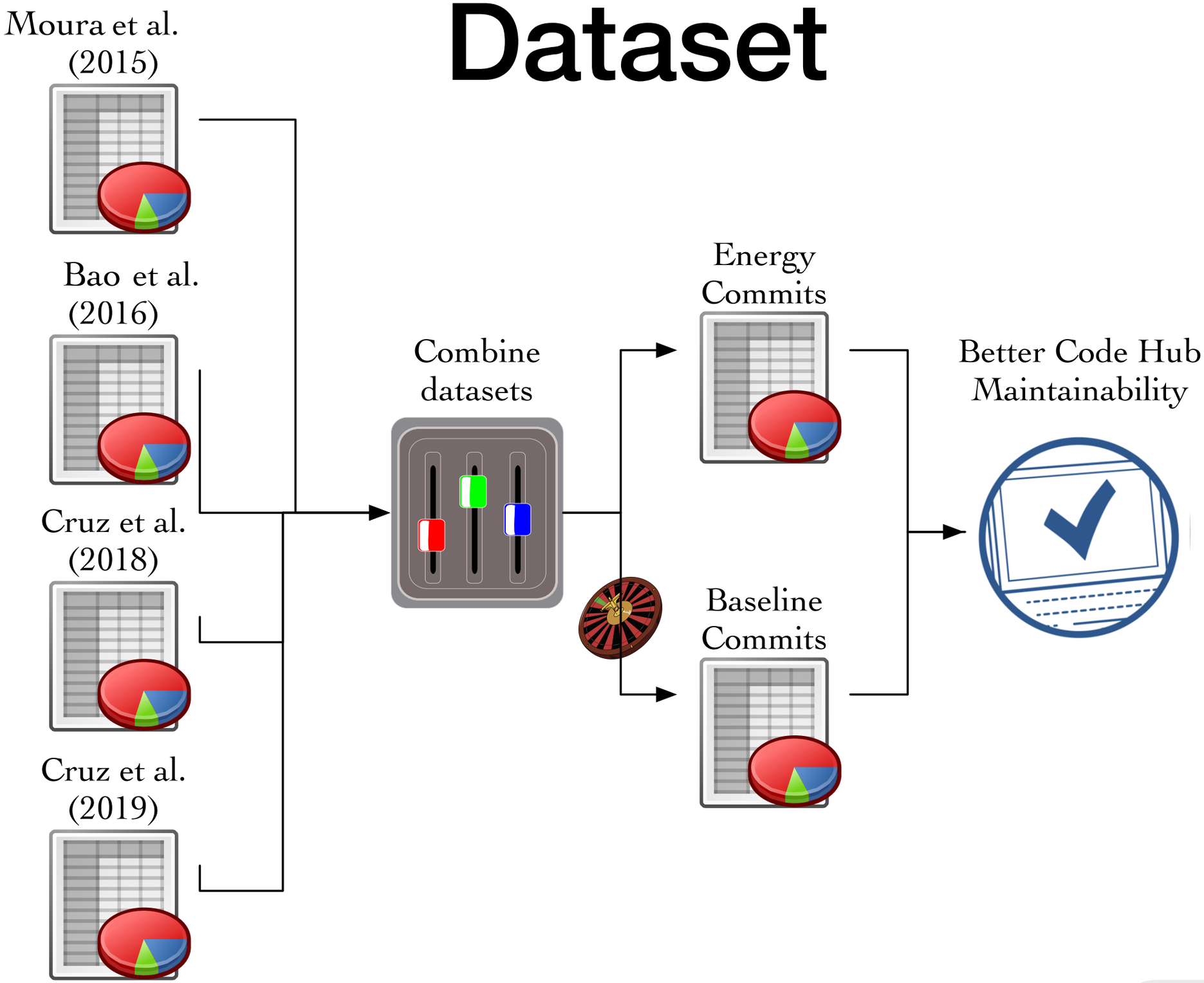
Numerical Score of Maintainability for a Software Version



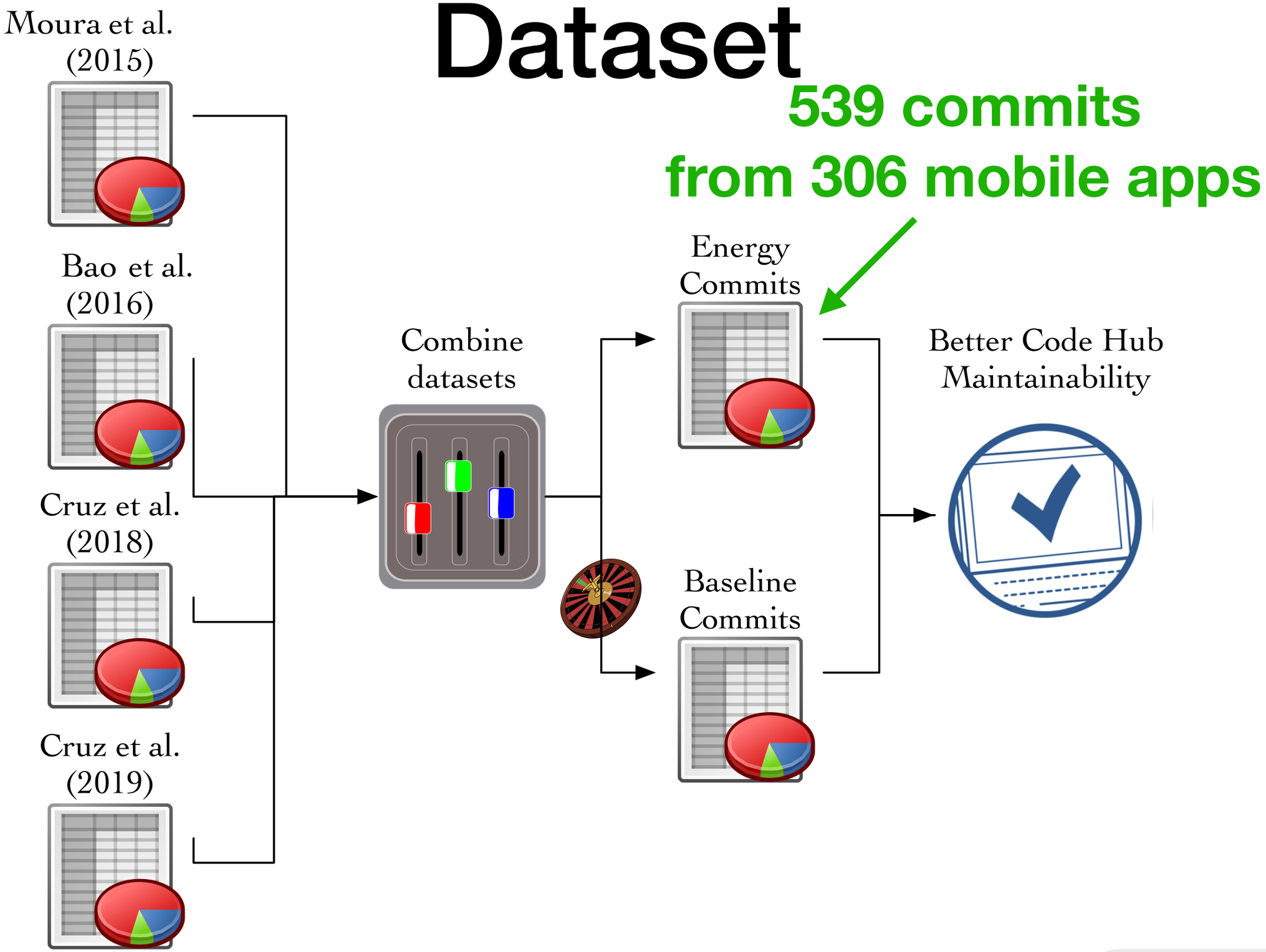
$$Maintainability = \frac{1}{3}\Bigg(SLOC_{green} - \frac{1-T_{yellow}}{T_{yellow}}(SLOC_{yellow} + SLOC_{orange} + SLOC_{red}) +$$
$$+SLOC_{green} + SLOC_{yellow} - \frac{1-T_{orange}}{T_{orange}}(SLOC_{orange} + SLOC_{red}) +$$
$$+SLOC_{green} + SLOC_{yellow} + SLOC_{orange} - \frac{1-T_{red}}{T_{red}}SLOC_{red}\Bigg)$$

Explained in the paper

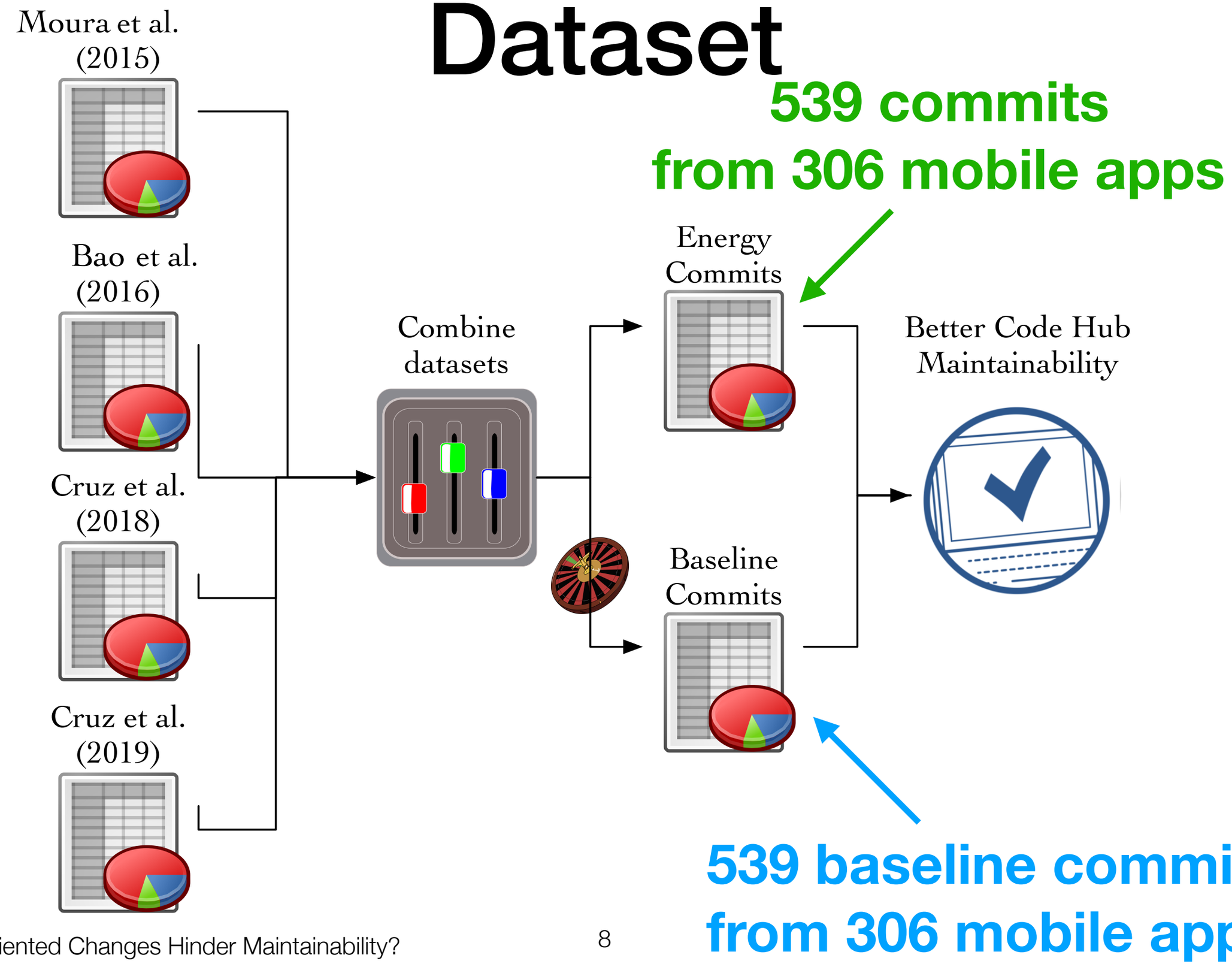
Energy Code Changes Dataset



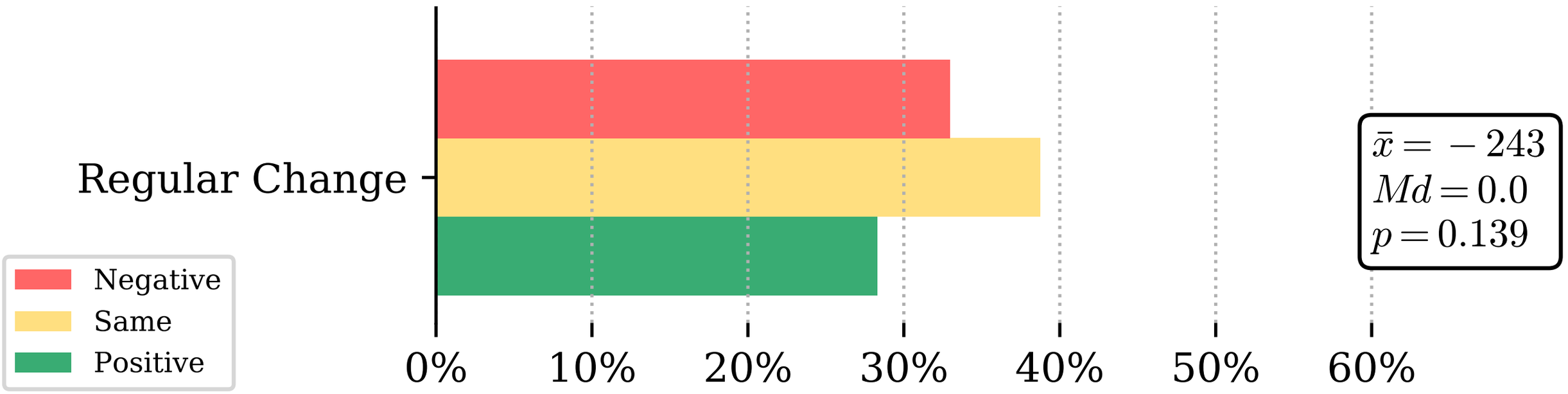
Energy Code Changes Dataset



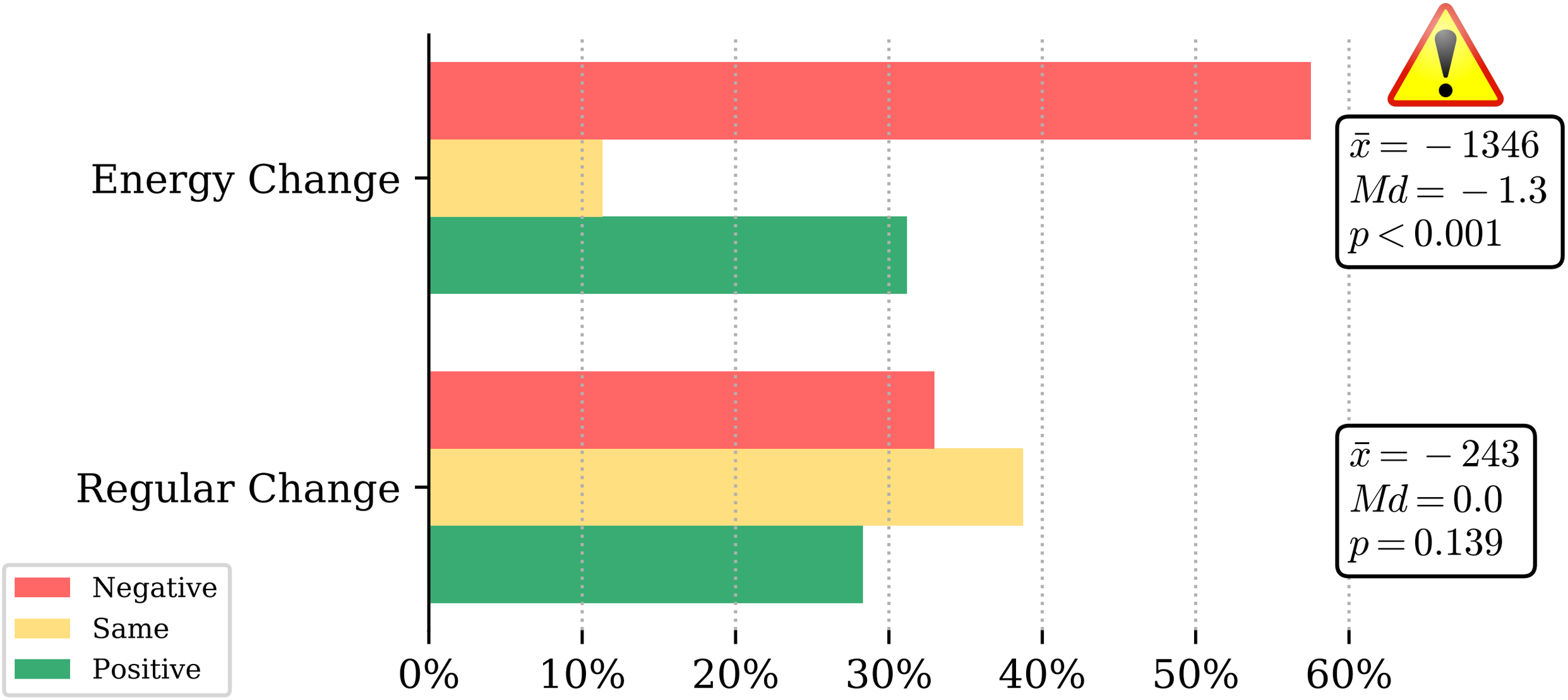
Energy Code Changes Dataset



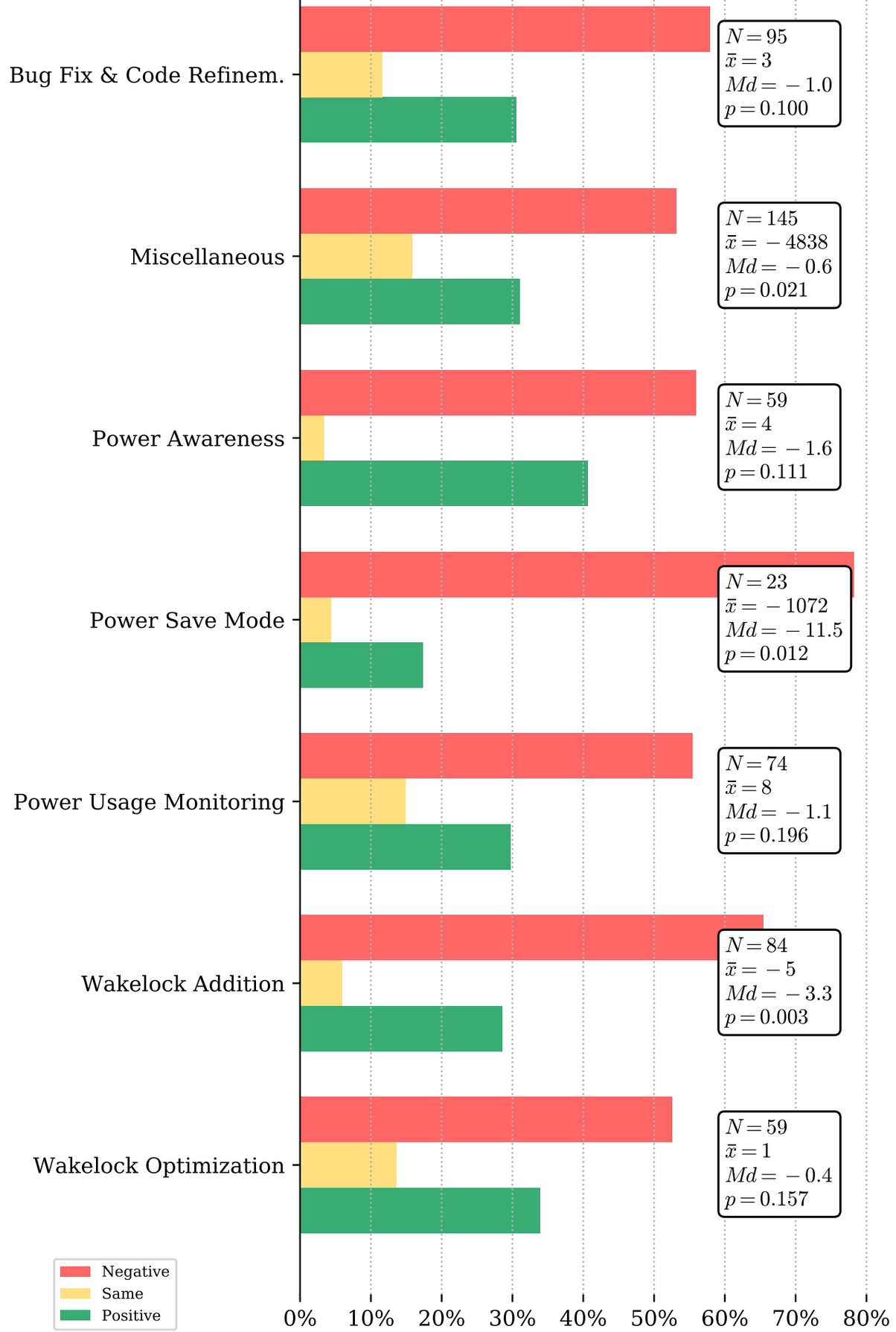
Impact of energy changes on maintainability



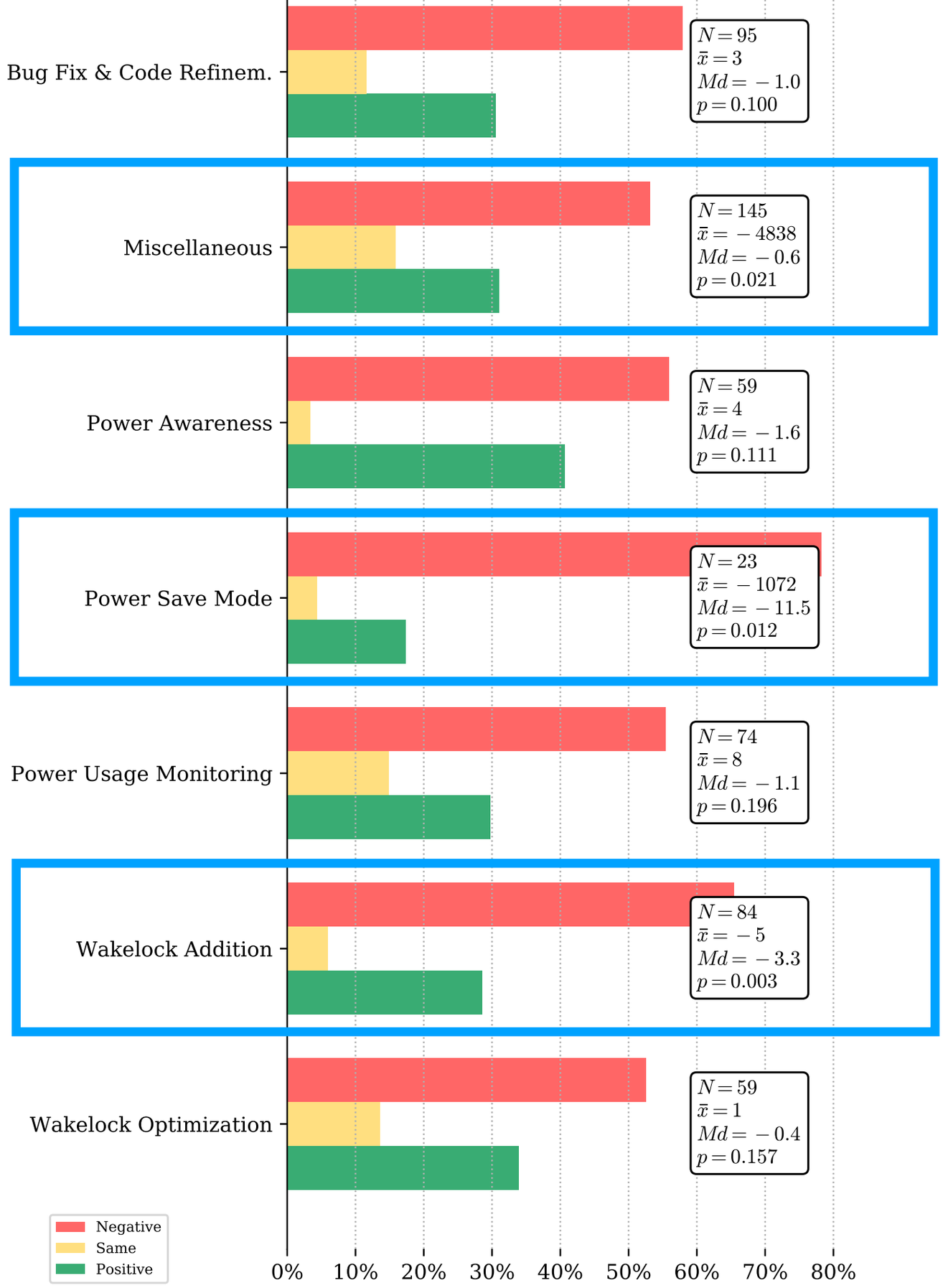
Impact of energy changes on maintainability



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;
    ...
```

Typical maintainability issue II

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

5 changed files with **66 additions** and **14 deletions**.

```
+
+                // each time we reconnect, check to see if we're suppose to be
+                // in power saving mode.  if not, start the scanning. TODO: we
+                // shouldn't just stopScanning if we find that we are in PSM.
+                // Instead, we should see if we were scanning do to an activity
+                // recognition.  If we were, don't stop.
+                if (mConnectionRemote != null) {
+                    try {
+                        if (mPrefs.getPowerSavingMode()) {
+                            mConnectionRemote.stopScanning();
+                        } else {
+                            mConnectionRemote.startScanning();
+                        }
+                    } catch (RemoteException e) {
+                        Log.e(LOGTAG, "", e);
+                    }
+                }
                 updateUI();
             }
```

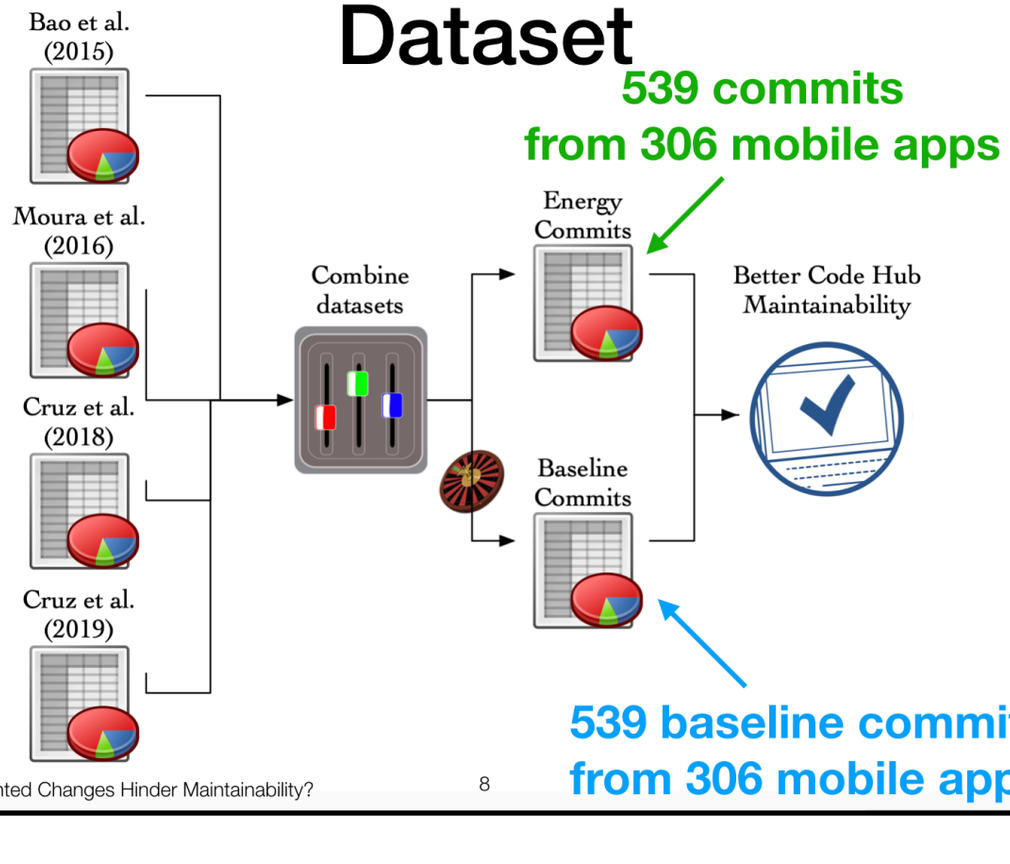
Conclusions

- 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



539 commits from 306 mobile apps

539 baseline commits from 306 mobile apps

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;
...
```


Which energy patterns are more likely to affect maintainability?