

Development

By The Numbers

$$3a(y+z)^2 +$$

$$(3y + (4 + 4A)(x +$$

$$\frac{a^2 C^3}{39}$$

$$(y+A)^{13} + \frac{2}{3}$$

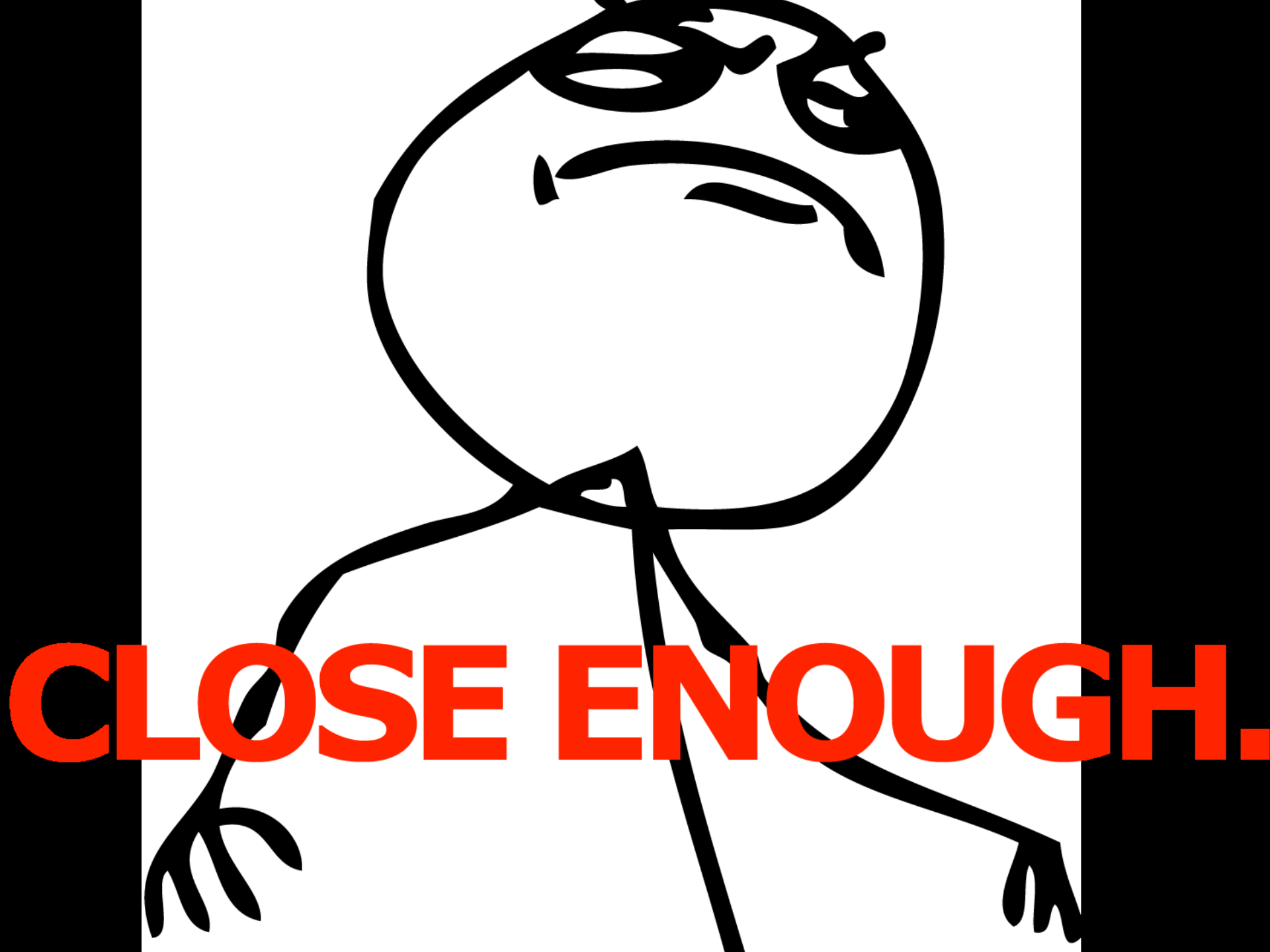
$$(A)^8 (x +$$

$$(x +$$



$$1 + 1 = 3$$





CLOSE ENOUGH.

**We Are Going To
Measure Complexity**

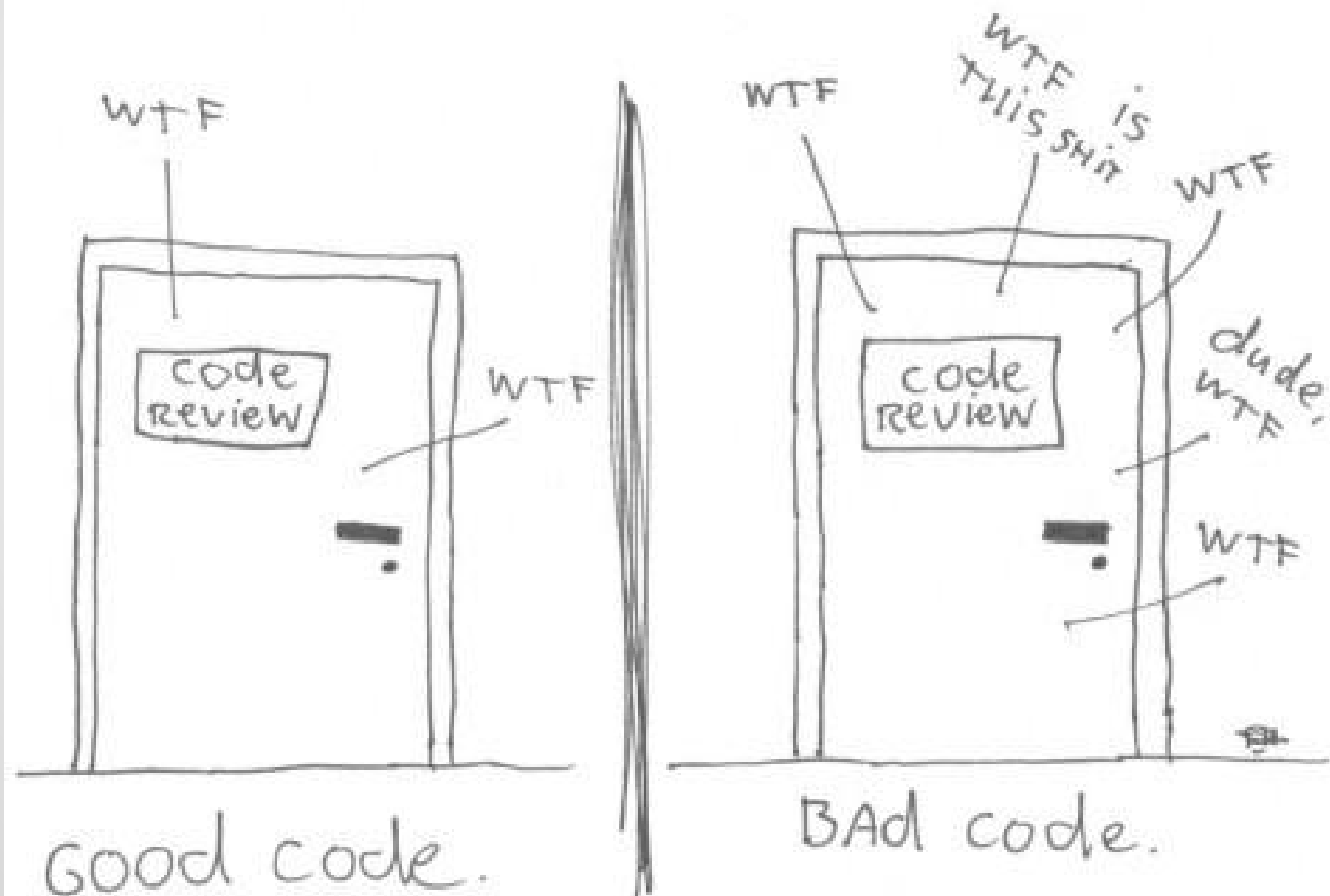
**Why Should We Care
About Complexity?**

***"The Central Enemy Of
Reliability is Complexity"***

- Geer et al.

**Complexity And
Quality Are Strongly
Related**

The ONLY valid measurement of code quality: WTFs/minute



Basic Metrics

Cyclomatic Complexity

Cyclomatic Complexity

Number Of
"Decision Points"
In A Routine

```
function foo ($a, $b) {  
    $c = 0;  
    if ($a) {  
        $c = $a;  
    } elseif ($b) {  
        $c = $b;  
    }  
    if ($a && $b) {  
        $c = $a + $b;  
    }  
    return $c;  
}
```

```
function foo ($a, $b) {
```

```
    $c = 0;
```

```
    if ($a) {
```

```
        $c = $a;
```

```
    } elseif ($b) {
```

```
        $c = $b;
```

```
    }
```

```
    if ($a && $b) {
```

```
        $c = $a + $b;
```

```
    }
```

```
    return $c;
```

```
}
```

```
function foo ($a, $b) {
```

```
    $c = 0;
```

```
    if ($a) {
```

```
        $c = $a;
```

```
    } elseif ($b) {
```

```
        $c = $b;
```

```
    }
```

```
    if ($a && $b) {
```

```
        $c = $a + $b;
```

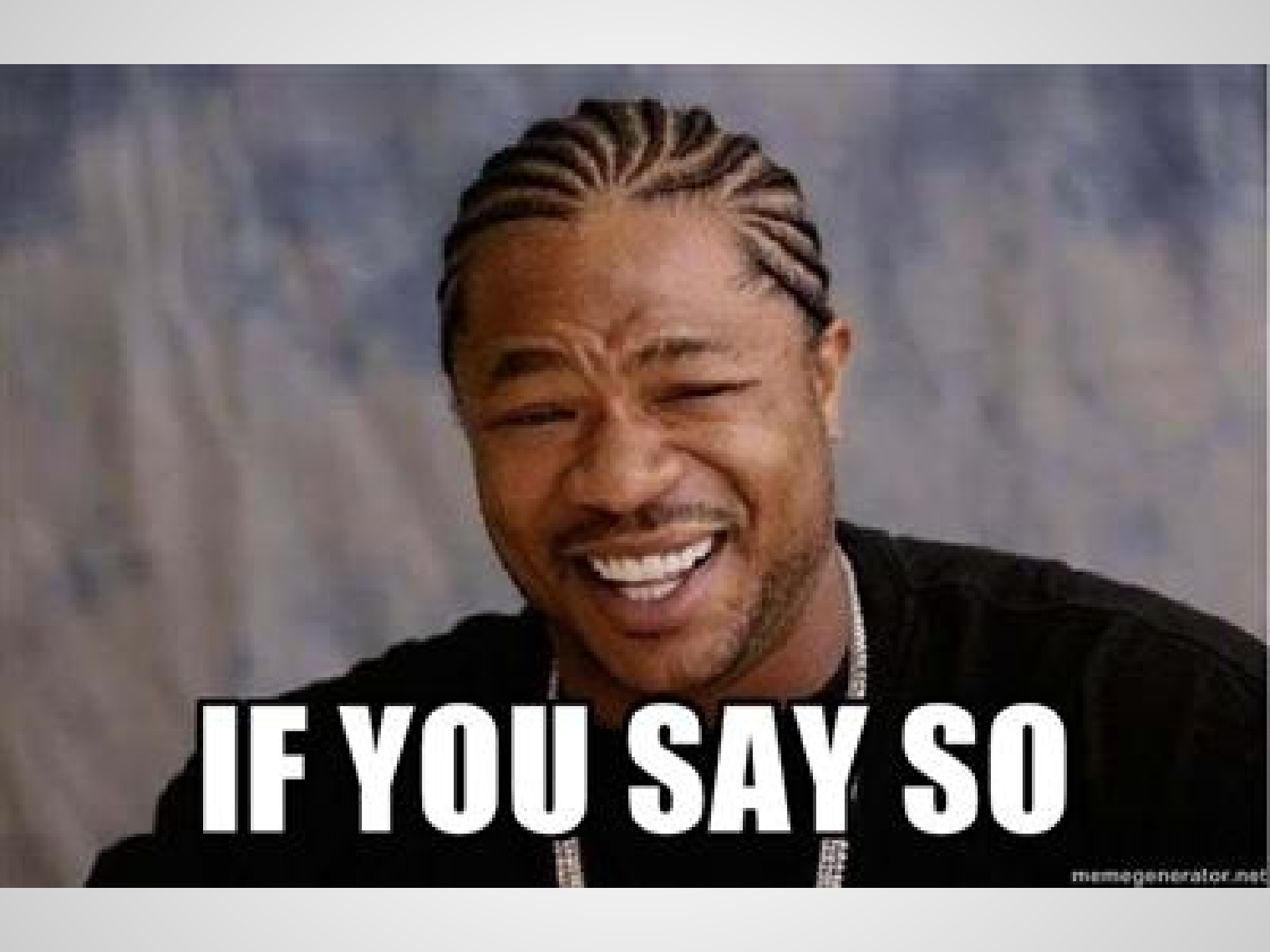
```
    }
```

```
    return $c;
```

```
}
```

4

**Simple
Right?**



IF YOU SAY SO

Cyclomatic Complexity (Single Method)

1 - 4: Low Complexity

5 - 7: Moderate Complexity

8 - 10: High Complexity

11+: Very High Complexity

Cyclomatic Complexity (Average Per Method)

- 1 - 2:** Low Complexity
- 2 - 4:** Moderate Complexity
- 4 - 6:** High Complexity
- 6+:** Very High Complexity

Compare:

Average CC per Method

Wordpress:	6.28
Drupal 7:	3.02
Drupal 8:	2.10
Symfony 2:	1.81
Zend Framework 2:	2.62
Laravel:	1.79

Cyclomatic Complexity (Average Per Line Of Code)

.01 - .05: Low Complexity

.05 - .10: Moderate Complexity

.10 - .15: High Complexity

.15+: Very High Complexity

Compare:

Average CC per LOC

Wordpress:	0.20
Drupal 7:	0.04
Drupal 8:	0.07
Symfony 2:	0.06
Zend Framework 2:	0.10
Laravel:	0.07

NOT SURE IF GOOD THING



OR BAD THING

N-Path Complexity


N-Path Complexity

Number Of
"Unique Paths"
In A Routine

```
function foo ($a, $b) {  
    $c = 0;  
    if ($a) {  
        $c = $a;  
    } elseif ($b) {  
        $c = $b;  
    }  
    if ($a && $b) {  
        $c = $a + $b;  
    }  
    return $c;  
}
```

```
function foo ($a, $b) {  
    $c = 0;  
    if ($a) {  
        $c = $a;  
    } elseif ($b) {  
        $c = $b;  
    }  
    if ($a && $b) {  
        $c = $a + $b;  
    }  
    return $c;  
}
```

```
function foo ($a, $b) {  
    $c = 0;  
    if ($a) {  
        $c = $a;  
    } elseif ($b) {  
        $c = $b;  
    }  
    if ($a && $b) {  
        $c = $a + $b;  
    }  
    return $c;  
}
```



```
function foo ($a, $b) {
```

```
    $c = 0;
```

```
    if ($a) {
```

```
        $c = $a;
```

```
    } elseif ($b) {
```

```
        $c = $b;
```

```
    }
```

```
    if ($a && $b) {
```

```
        $c = $a + $b;
```

```
    }
```

```
    return $c;
```

```
}
```

```
function foo ($a, $b) {
```

```
    $c = 0;
```

```
    if ($a) {
```

```
        $c = $a;
```

```
    } elseif ($b) {
```

```
        $c = $b;
```

```
    }
```

```
    if ($a && $b) {
```

```
        $c = $a + $b;
```

```
    }
```

```
    return $c;
```

```
}
```

```
function foo ($a, $b) {
```

```
    $c = 0;
```

```
    if ($a) {
```

```
        $c = $a;
```

```
    } elseif ($b) {
```

```
        $c = $b;
```

```
    }
```

```
    if ($a && $b) {
```

```
        $c = $a + $b;
```

```
    }
```

```
    return $c;
```

4

```
}
```


**They Are
The Same?**

**Not
Generally!**

```
function foo2 ($a, $b, $c) {  
    $d = 0;  
    if ($a) {  
        $d += $a;  
    }  
    if ($b) {  
        $d += $b;  
    }  
    if ($c) {  
        $d += $c;  
    }  
    return $d;  
}
```

CC:
NPath:

```
function foo2 ($a, $b, $c) {  
    $d = 0;  
    if ($a) {  
        $d += $a;  
    }  
    if ($b) {  
        $d += $b;  
    }  
    if ($c) {  
        $d += $c;  
    }  
    return $d;  
}
```

CC:
NPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4

NPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4
NPath:

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```

$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}

```

CC: NPath

```
$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}
```

```
$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}
```

```
$d = 0;
if ($a) {
    $d += $a;
}
if ($b) {
    $d += $b;
}
if ($c) {
    $d += $c;
}
return $d;
}
```

CC: 4

NPPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4

NPath:


```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4
NPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4
NPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4
NPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4

NPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4
NPath:

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4

NPath: 8

```
function foo2 ($a, $b, $c) {
```

```
    $d = 0;
```

```
    if ($a) {
```

```
        $d += $a;
```

```
    }
```

```
    if ($b) {
```

```
        $d += $b;
```

```
    }
```

```
    if ($c) {
```

```
        $d += $c;
```

```
    }
```

```
    return $d;
```

```
}
```

CC: 4

NPath: 8

$2^{(CC-1)}$

N-Path Complexity

<16: Low Complexity

17-128: Moderate Complexity

129-1024: High Complexity

1025+: Very High Complexity



N-Path Complexity

**Minimum Number Of
Tests Required To
Completely Test
A Routine**



```
function entity_load($entity_type, $ids = FALSE, $conditions = array(), $reset = FALSE) {  
  if ($reset) {  
    entity_get_controller($entity_type)->resetCache();  
  }  
  return entity_get_controller($entity_type)->load($ids, $conditions);  
}
```

N-Path Complexity

`entity_load()`

CC:

N-Path:

N-Path Complexity

`entity_load()`

CC: 2

N-Path:

Cyclomatic Complexity

1 - 4: Low Complexity

5 - 7: Moderate Complexity

8 - 10: High Complexity

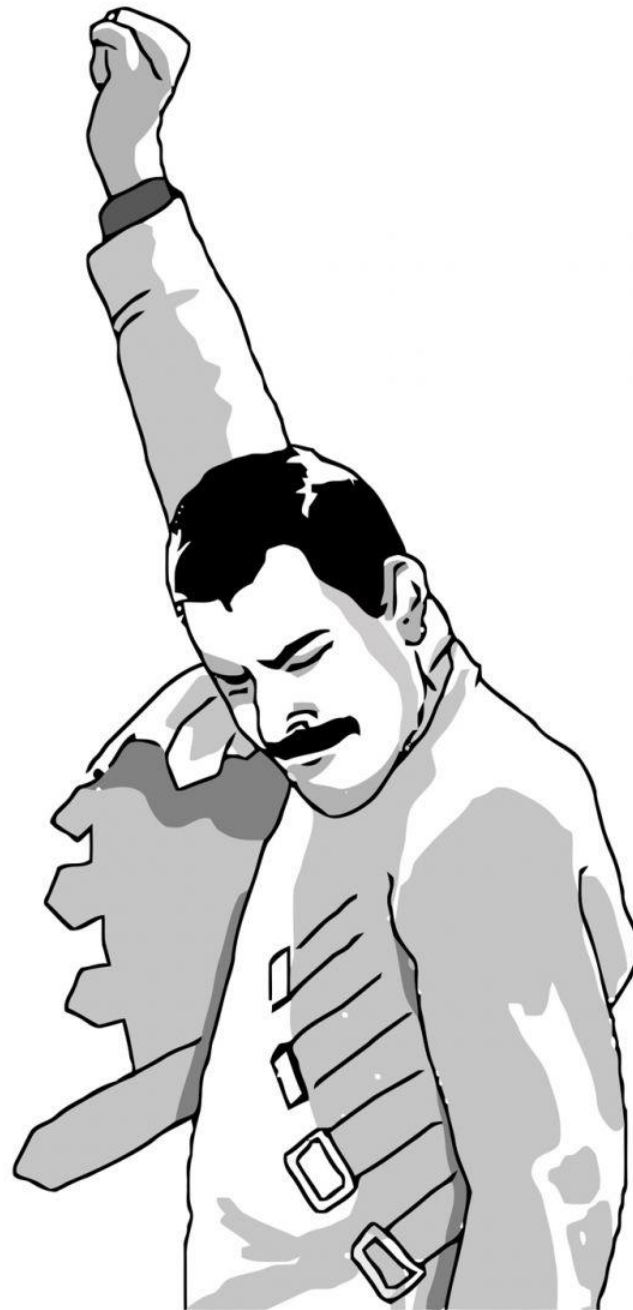
11+: Very High Complexity

N-Path Complexity

`entity_load()`

CC: 2

N-Path: 2



N-Path Complexity

`drupal_http_request()`

CC:

N-Path:

```

function drupal_http_request($url, array $options = array()) {
    // Allow an alternate HTTP client library to replace Drupal's default
    // implementation.
    $override_function = variable_get('drupal_http_request_function', FALSE);
    if (!empty($override_function) && function_exists($override_function)) {
        return $override_function($url, $options);
    }

    $result = new stdClass();

    // Parse the URL and make sure we can handle the schema.
    $uri = @parse_url($url);

    if ($uri == FALSE) {
        $result->error = 'unable to parse URL';
        $result->code = -1001;
        return $result;
    }

    if (!isset($uri['scheme'])) {
        $result->error = 'missing schema';
        $result->code = -1002;
        return $result;
    }

    timer_start(__FUNCTION__);

    // Merge the default options.
    $options += array(
        'headers' => array(),
        'method' => 'GET',
        'data' => NULL,
        'max_redirects' => 3,
        'timeout' => 30.0,
        'context' => NULL,
    );

    // Merge the default headers.
    $options['headers'] += array(
        'User-Agent' => 'Drupal (+http://drupal.org/)',
    );

    // stream_socket_client() requires timeout to be a float.
    $options['timeout'] = (float) $options['timeout'];

    // Use a proxy if one is defined and the host is not on the excluded list.
    $proxy_server = variable_get('proxy_server', '');
    if ($proxy_server && !_drupal_http_use_proxy($uri['host'])) {
        // Set the scheme so we open a socket to the proxy server.
        $uri['scheme'] = 'proxy';
        // Set the path to be the full URL.
        $uri['path'] = $url;
        // Since the URL is passed as the path, we won't use the parsed query.
        unset($uri['query']);

        // Add in username and password to Proxy-Authorization header if needed.
        if ($proxy_username = variable_get('proxy_username', '')) {
            $proxy_password = variable_get('proxy_password', '');
            $options['headers']['Proxy-Authorization'] = 'Basic ' . base64_encode($proxy_username . (!empty($proxy_password) ? ':' . $proxy_password : ''));
        }

        // Some proxies reject requests with any User-Agent headers, while others
        // require a specific one.
        $proxy_user_agent = variable_get('proxy_user_agent', '');
        // The default value matches neither condition.
        if ($proxy_user_agent === NULL) {
            unset($options['headers']['User-Agent']);
        }
        elseif ($proxy_user_agent) {
            $options['headers']['User-Agent'] = $proxy_user_agent;
        }
    }

    switch ($uri['scheme']) {
        case 'proxy':
            // Make the socket connection to a proxy server.
            $socket = 'tcp://' . $proxy_server . ':' . variable_get('proxy_port', 8080);
            // The Host header still needs to match the real request.
            $options['headers']['Host'] = $uri['host'];
            $options['headers']['Host'] .= isset($uri['port']) && $uri['port'] != 80 ? ':' . $uri['port'] : '';
            break;

        case 'http':
        case 'feed':
    }

```

```

$port = isset($uri['port']) ? $uri['port'] : 80;
$socket = 'tcp://' . $uri['host'] . ':' . $port;
// RFC 2616: "non-standard ports MUST, default ports MAY be included".
// We don't add the standard port to prevent from breaking rewrite rules
// checking the host that do not take into account the port number.
$options['headers']['Host'] = $uri['host'] . ($port != 80 ? ':' . $port : '');
break;

case 'https':
    // Note: Only works when PHP is compiled with OpenSSL support.
    $port = isset($uri['port']) ? $uri['port'] : 443;
    $socket = 'ssl://' . $uri['host'] . ':' . $port;
    $options['headers']['Host'] = $uri['host'] . ($port != 443 ? ':' . $port : '');
    break;

default:
    $result->error = 'invalid schema ' . $uri['scheme'];
    $result->code = -1003;
    return $result;
}

if (empty($options['context'])) {
    $fp = @stream_socket_client($socket, $errno, $errstr, $options['timeout']);
}
else {
    // Create a stream with context. Allows verification of a SSL certificate.
    $fp = @stream_socket_client($socket, $errno, $errstr, $options['timeout'], STREAM_CLIENT_CONNECT, $options['context']);
}

// Make sure the socket opened properly.
if (!$fp) {
    // When a network error occurs, we use a negative number so it does not
    // clash with the HTTP status codes.
    $result->code = -$errno;
    $result->error = trim($errstr) ? trim($errstr) : t('Error opening socket @socket', array('@socket' => $socket));

    // Mark that this request failed. This will trigger a check of the web
    // server's ability to make outgoing HTTP requests the next time that
    // requirements checking is performed.
    // See system_requirements().
    variable_set('drupal_http_request_fails', TRUE);

    return $result;
}

// Construct the path to act on.
$path = isset($uri['path']) ? $uri['path'] : '/';
if (isset($uri['query'])) {
    $path .= '?' . $uri['query'];
}

// Only add Content-Length if we actually have any content or if it is a POST
// or PUT request. Some non-standard servers get confused by Content-Length in
// at least HEAD/GET requests, and Squid always requires Content-Length in
// POST/PUT requests.
$content_length = strlen($options['data']);
if ($content_length > 0 || $options['method'] == 'POST' || $options['method'] == 'PUT') {
    $options['headers']['Content-Length'] = $content_length;
}

// If the server URL has a user then attempt to use basic authentication.
if (isset($uri['user'])) {
    $options['headers']['Authorization'] = 'Basic ' . base64_encode($uri['user'] . (isset($uri['pass']) ? ':' . $uri['pass'] : ''));
}

// If the database prefix is being used by SimpleTest to run the tests in a copied
// database then set the user-agent header to the database prefix so that any
// calls to other Drupal pages will run the SimpleTest prefixed database. The
// user-agent is used to ensure that multiple testing sessions running at the
// same time won't interfere with each other as they would if the database
// prefix were stored statically in a file or database variable.
$test_info = &$GLOBALS['drupal_test_info'];
if (empty($test_info['test_run_id'])) {
    $options['headers']['User-Agent'] = drupal_generate_test_ua($test_info['test_run_id']);
}

$request = $options['method'] . ' ' . $path . " HTTP/1.0\r\n";
foreach ($options['headers'] as $name => $value) {
    $request .= $name . ': ' . trim($value) . "\r\n";
}
$request .= "\r\n" . $options['data'];
$result->request = $request;
// Calculate how much time is left of the original timeout value.
$timeout = $options['timeout'] - timer_read(__FUNCTION__) / 1000;
if ($timeout > 0) {
    stream_set_timeout($fp, floor($timeout), floor(1000000 * fmod($timeout, 1)));
    fwrite($fp, $request);
}

```

```

// Fetch response. Due to PHP bugs like http://bugs.php.net/bug.php?id=43782
// and http://bugs.php.net/bug.php?id=46049 we can't rely on feof(), but
// instead must invoke stream_get_meta_data() each iteration.
$info = stream_get_meta_data($fp);
$alive = !$info['eof'] && !$info['timed_out'];
$response = '';

while ($alive) {
    // Calculate how much time is left of the original timeout value.
    $timeout = $options['timeout'] - timer_read(__FUNCTION__) / 1000;
    if ($timeout <= 0) {
        $info['timed_out'] = TRUE;
        break;
    }
    stream_set_timeout($fp, floor($timeout), floor(1000000 * fmod($timeout, 1)));
    $chunk = fread($fp, 1024);
    $response .= $chunk;
    $info = stream_get_meta_data($fp);
    $alive = !$info['eof'] && !$info['timed_out'] && $chunk;
}
fclose($fp);

if ($info['timed_out']) {
    $result->code = HTTP_REQUEST_TIMEOUT;
    $result->error = 'request timed out';
    return $result;
}

// Parse response headers from the response body.
// Be tolerant of malformed HTTP responses that separate header and body with
// \n\n or \r\r instead of \r\n\r\n.
list($response, $result->data) = preg_split("/\r\n\r\n\r\n\r\n\r\n/", $response, 2);
$response = preg_split("/\r\n\r\n\r\n\r\n/", $response);

// Parse the response status line.
list($protocol, $code, $status_message) = explode(' ', trim(array_shift($response)), 3);
$result->protocol = $protocol;
$result->status_message = $status_message;

$result->headers = array();

// Parse the response headers.
while ($line = trim(array_shift($response))) {
    list($name, $value) = explode(':', $line, 2);
    $name = strtolower($name);
    if (isset($result->headers[$name]) && $name == 'set-cookie') {
        // RFC 2109: the Set-Cookie response header comprises the token Set-
        // Cookie:, followed by a comma-separated list of one or more cookies.
        $result->headers[$name] .= ',' . trim($value);
    }
    else {
        $result->headers[$name] = trim($value);
    }
}

$responses = array(
    100 => 'Continue',
    101 => 'Switching Protocols',
    200 => 'OK',
    201 => 'Created',
    202 => 'Accepted',
    203 => 'Non-Authoritative Information',
    204 => 'No Content',
    205 => 'Reset Content',
    206 => 'Partial Content',
    300 => 'Multiple Choices',
    301 => 'Moved Permanently',
    302 => 'Found',
    303 => 'See Other',
    304 => 'Not Modified',
    305 => 'Use Proxy',
    307 => 'Temporary Redirect',
    400 => 'Bad Request',
    401 => 'Unauthorized',
    402 => 'Payment Required',
    403 => 'Forbidden',
    404 => 'Not Found',
    405 => 'Method Not Allowed',
    406 => 'Not Acceptable',
    407 => 'Proxy Authentication Required',
    408 => 'Request Time-out',
    409 => 'Conflict',
    410 => 'Gone',
    411 => 'Length Required',
    412 => 'Precondition Failed',
    413 => 'Request Entity Too Large',
    414 => 'Request-URI Too Large',
    415 => 'Unsupported Media Type',

```

```

416 => 'Requested range not satisfiable',
417 => 'Expectation Failed',
500 => 'Internal Server Error',
501 => 'Not Implemented',
502 => 'Bad Gateway',
503 => 'Service Unavailable',
504 => 'Gateway Time-out',
505 => 'HTTP Version not supported',
);
// RFC 2616 states that all unknown HTTP codes must be treated the same as the
// base code in their class.
if (!isset($responses[$code])) {
    $code = floor($code / 100) * 100;
}
$result->code = $code;

switch ($code) {
    case 200: // OK
    case 304: // Not modified
        break;
    case 301: // Moved permanently
    case 302: // Moved temporarily
    case 307: // Moved temporarily
        $location = $result->headers['location'];
        $options['timeout'] -= timer_read(__FUNCTION__) / 1000;
        if ($options['timeout'] <= 0) {
            $result->code = HTTP_REQUEST_TIMEOUT;
            $result->error = 'request timed out';
        }
        elseif ($options['max_redirects']) {
            // Redirect to the new location.
            $options['max_redirects']--;
            $result = drupal_http_request($location, $options);
            $result->redirect_code = $code;
        }
        if (!isset($result->redirect_url)) {
            $result->redirect_url = $location;
        }
        break;
    default:
        $result->error = $status_message;
}

return $result;
}

```

N-Path Complexity

`drupal_http_request()`

CC: 41

N-Path:

Cyclomatic Complexity

1 - 4: Low Complexity

5 - 7: Moderate Complexity

8 - 10: High Complexity

11+: Very High Complexity

N-Path Complexity

`drupal_http_request()`

CC: 41

N-Path: 25,303,344,960

WELL THERE'S YOUR PROBLEM



To Completely Test
`drupal_http_request()`
At 1 Line Of Code Per Test
Would Require
2 Terabytes
Worth Of Tests

To Completely Test
`drupal_http_request()`
At 1 Line Of Code Per Test
Would Require
412 DVD's
Worth Of Tests

To Completely Test

`drupal_http_request()`

At 1 Line Of Code Per Test

Would Require

670k Drupals

Worth Of Tests

**And That's Not
The Worst One!**

SAY

WHAT NOW?

N-Path Complexity

```
_date_repeat_rrule_process()
```

CC:

N-Path:

N-Path Complexity

```
_date_repeat_rrule_process()
```

CC: 81

N-Path:

N-Path Complexity

`_date_repeat_rrule_process()`

CC: 81

N-Path: 19,781,719,256

N-Path Complexity

`_date_repeat_rrule_process()`

CC: 81

N-Path: 19,781,719,256
,250,000,000,000

N-Path Complexity

`_date_repeat_rrule_process()`

CC: 81

N-Path: 19,781,719,256
,250,000,000,000
,000,000,000

To Completely Test

`_date_repeat_rrule_process()`

At 1 Line Of Code Per Test

Would Require

336T 2009's

Worth Of Tests

SanDisk
Mobile *Ultra*

64 GB micro
SD
XC I

MicroSD Card

- About 387GB / cm³
 - Current Density Record

MicroSD Card

- About 387GB / cm³
 - Current Density Record
- About 184PB / Butt-Load
 - The text from every piece of written material in the world...

MicroSD Card

- About 387GB / cm³
 - Current Density Record
- About 184PB / Butt-Load
 - The text from every piece of written material in the world...
- About 387PB / m³
 - About 20 “MegaUpload.com Sites”

MicroSD Card

- About 387GB / cm³
 - Current Density Record
- About 184PB / Butt-Load
 - The text from every piece of written material in the world...
- About 387PB / m³
 - About 20 “MegaUpload.com Sites”
- About 387YB / km³
 - About 100,000 Google’s

To Completely Test

`_date_repeat_rrule_process()`

At 1 Line Of Code Per Test

Would Require

1 Greenland Ice Cap of

microSD cards

Worth Of Tests



NOT BAD

N-Path Complexity

`compose_pm()`

CC:

N-Path:

N-Path Complexity

`compose_pm()`

CC: 331

N-Path:

N-Path Complexity

`compose_pm()`

CC: 331

N-Path: 12,334,686,665

N-Path Complexity

`compose_pm()`

CC: 331

N-Path: 12,334,686,665
,165,904,892,028

N-Path Complexity

`compose_pm()`

CC: 331

N-Path: 12,334,686,665
,165,904,892,028
,680,955,918,269

N-Path Complexity

`compose_pm()`

CC: 331

N-Path: 12,334,686,665
,165,904,892,028
,680,955,918,269
,333,860,397,338

N-Path Complexity

`compose_pm()`

CC: 331

N-Path: 12,334,686,665

,165,904,892,028

,680,955,918,269

,333,860,397,338

,867,187,500,000

`compose_path()`

N-Path Complexity

CC: 331

N-Path: 12,334,686,665

,165,904,892,028

,680,955,918,269

,333,860,397,338

,867,187,500,000

,000,000,000,000

~~N-Path~~ Complexity

N-Path: 12,334,686,665

,165,904,892,028

,680,955,918,269

,333,860,397,338

,867,187,500,000

,000,000,000,000

,000,000,000,000

**Because Sending Private
Messages Is Hard...**

To Completely Test

`compose_pm()`

At 1 Line Of Code Per Test

Would Require

1B BlackHoles

Worth Of Tests

To Completely Test

`compose_pm()`

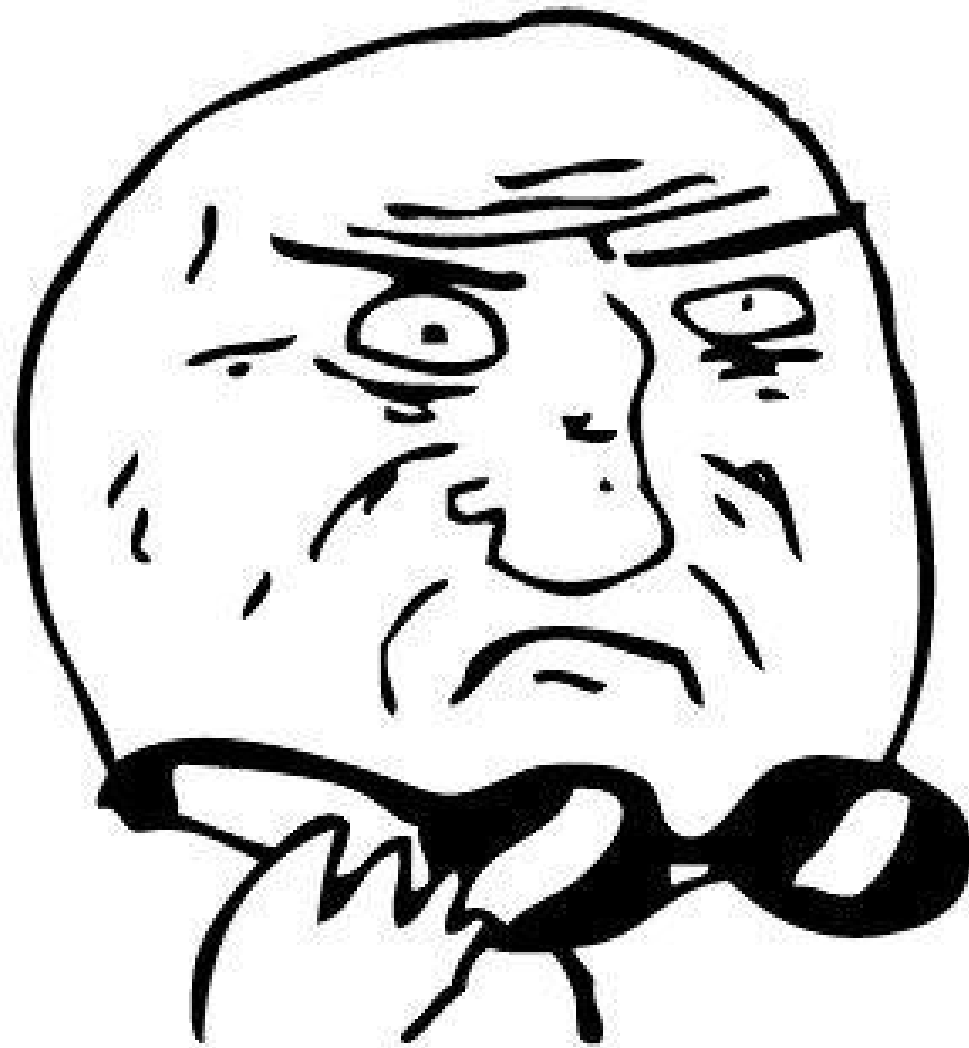
At 1 Line Of Code Per Test

Would Require

15,000,000 Milky Way's of

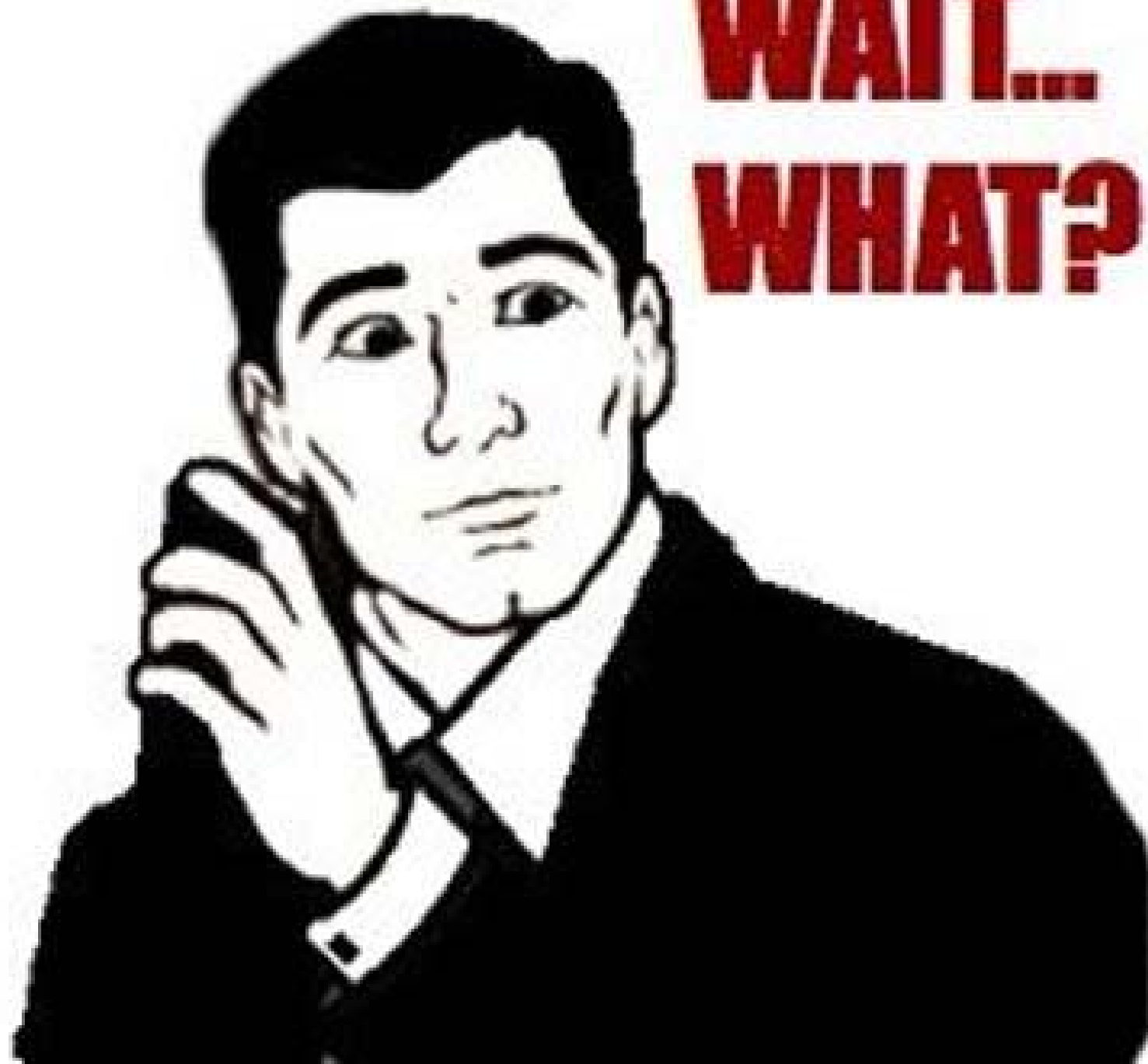
MicroSDs

Worth Of Tests



MOTHER OF GOD

CRAP



CRAP

(Change Risk Analysis Predictions)

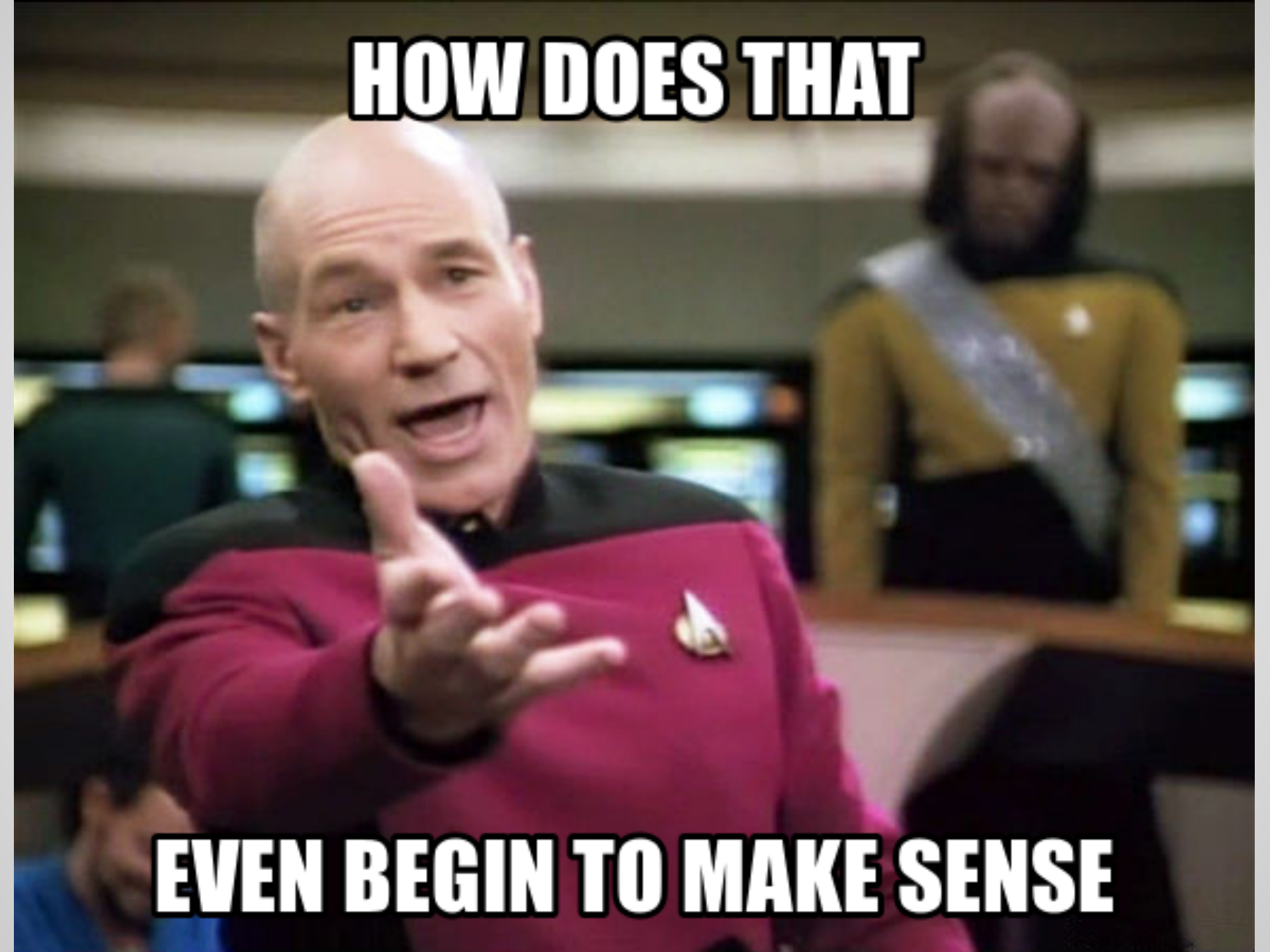
CC = Cyclomatic Complexity (method)

COV = Test Coverage (percent)

$CRAP = CC + (CC^2 * (1 - COV)^3)$

HOW DOES THAT

EVEN BEGIN TO MAKE SENSE



CRAP

**Relates Complexity
And Test Coverage**

CRAP

Increasing Test Coverage Lowers CRAP
Decreasing Complexity Lowers CRAP

CRAP

A Low Complexity Method
With No Tests

Is Good

CRAP

A Low Complexity Method
With Good Tests

Is Great

CRAP

A Moderate Complexity Method
With Good Tests

Is OK

CRAP

A Moderate Complexity Method
With No Tests

Is CRAP

CRAP

< 5: GREAT Code

5 - 15: Acceptable Code

15-30: Eih... Code

30+: CRAPpy Code



**How Do We
Apply These
Metrics?**



TEST



Quiz:

**Only 2 Major QA Tools
NOT Made By Germans**

What Are They?

PHP-CodeSniffer

Behat



Sebastian Bergmann

PHPUnit

DbUnit

PHPLOC

PHPCPD

PHPCOV

hphpa

www.phpqatools.org

www.jenkins-php.org

PHPLOC

PHPLOC

By Sebastian Bergmann

PHPLOC

By Sebastian Bergmann

Command Line Tool

PHPLOC

By Sebastian Bergmann

Command Line Tool

Summarizes An Entire
Codebase

```
$ phploc path/to/Drupal7/
```

```
Directories: 73
```

```
Files: 180
```

```
Lines of Code (LOC): 63347
```

```
    Cyclomatic Complexity / Lines of Code: 0.04
```

```
Comment Lines of Code (CLOC): 19321
```

```
Non-Comment Lines of Code (NCLOC): 44026
```

Namespaces: **0**

Interfaces: **1**

Traits: **0**

Classes: **38**

Abstract: **2** (5.26%)

Concrete: **36** (94.74%)

Average Class Length (NCLOC) : **197**

Methods: **433**

Scope:

Non-Static: **378** (87.30%)

Static: **55** (12.70%)

Visibility:

Public: **255** (58.89%)

Non-Public: **178** (41.11%)

Average Method Length (NCLOC) : **17**

Cyclomatic Complexity / Number of Methods: **3.02**

Anonymous Functions: **0**

Functions: **521**

Constants: **22**

Global constants: **15**

Class constants: **7**

PDepend

PDepend

By Manuel Pichler
(Also German)

PDepend

By Manuel Pichler
(Also German)

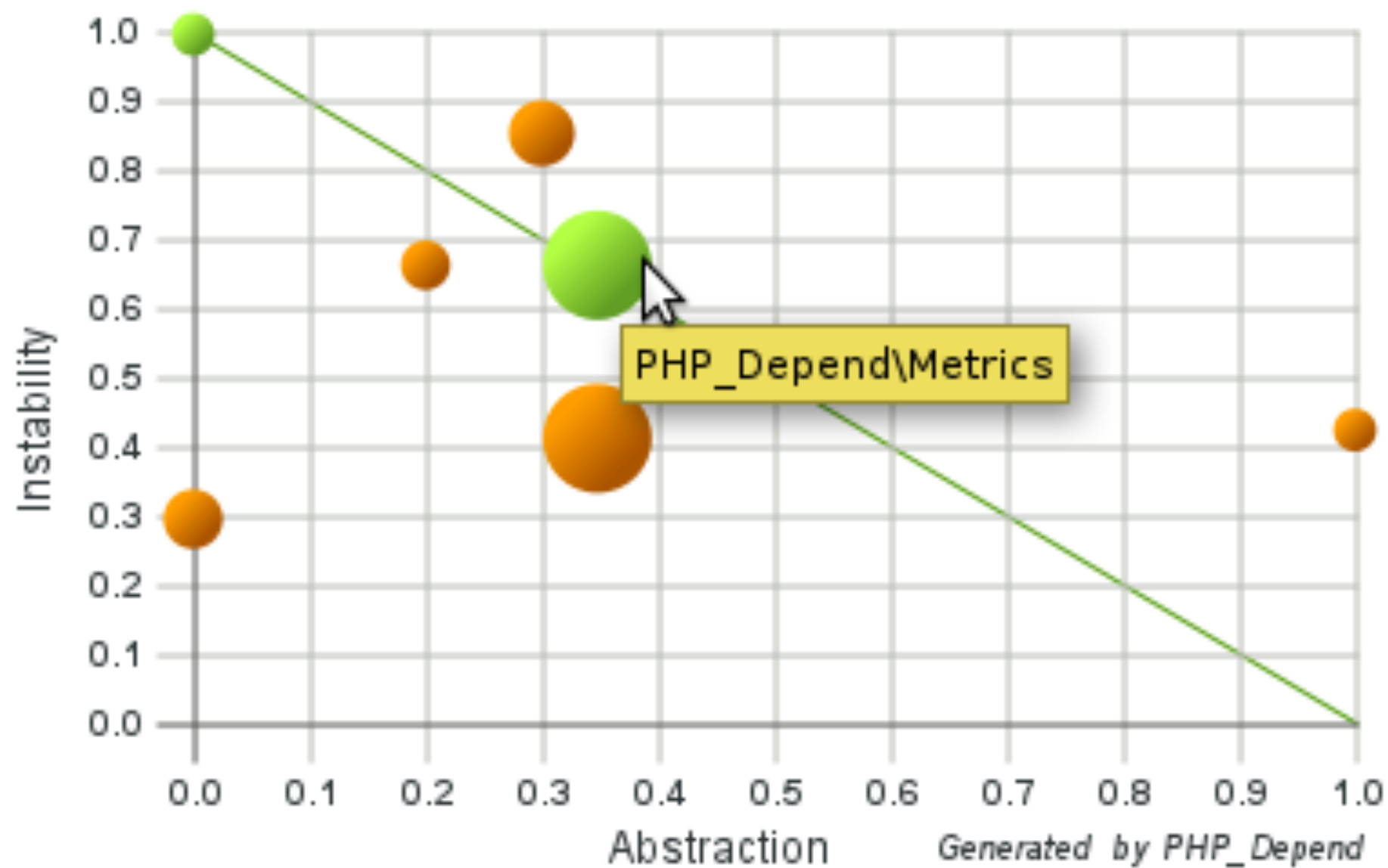
Like PHPLOC, But Granular

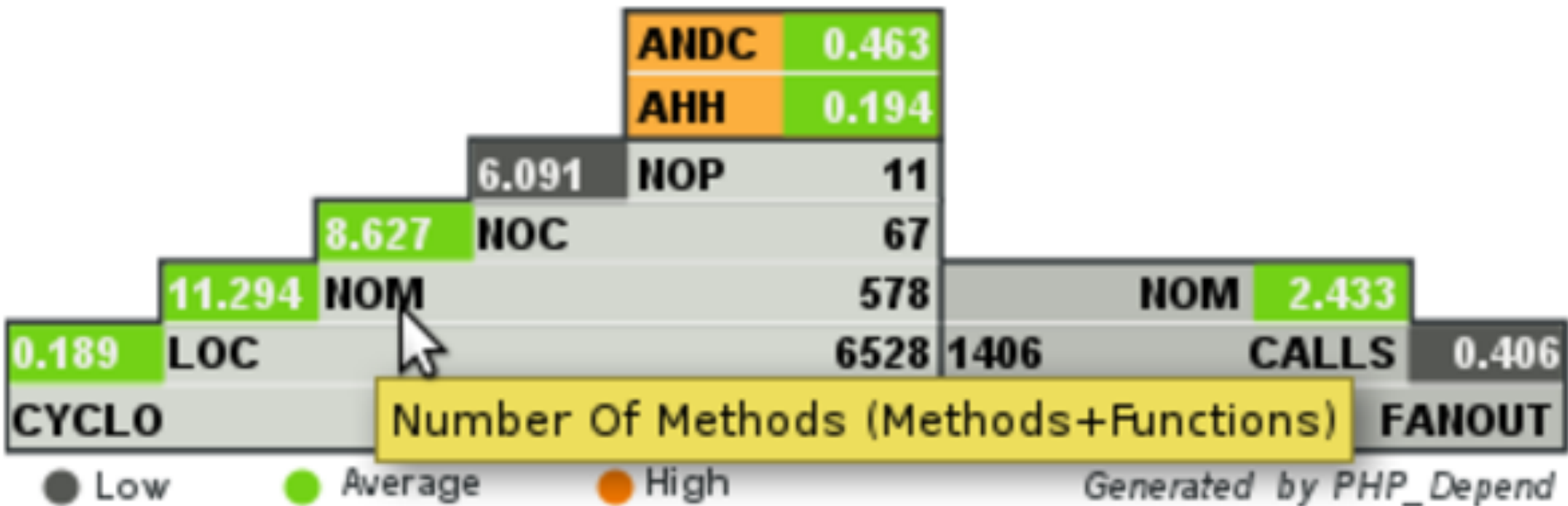
PDepend

By Manuel Pichler
(Also German)

Like PHPLOC, But Granular

Lower Level Analysis





Fanout: Describes Outward Dependencies

- Describes Dependence on Other Classes

ANDC: Average Num of Derived Classes

- Describes How Much Inheritance Is Used

AHH: Average Hierarchy Height

- Describes How Deep Of Inheritance Is Used

PHPMD

(Mess Detector)

PHPMD

By Manuel Pichler
(German)

PHPMD

By Manuel Pichler
(German)

Finds "Messy" Parts Of Code

PHPMD

By Manuel Pichler
(German)

Finds "Messy" Parts Of Code

Finds Rule Violations

PHPMD Rules

CodeSize

- (CC, NPath, Number of Methods, Size of Methods, etc)

Design

- (Eval, Goto, Exit(), Inheritance Depth)

Naming

- (Short names, Inconsistent Names)

Unused Code

Controversial

- (Superglobal Access, Naming Conventions)



WHAT WOULD YOU SAY

YOU'RE TRYING TO DO HERE?

Prevent Complex Code
From Even Getting In!

**By Themselves
Useful**

Over Time

**Over Time
Invaluable**

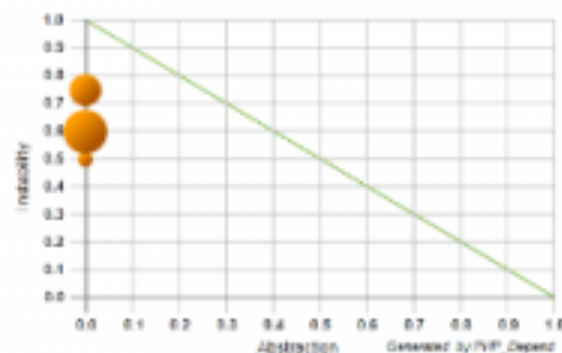
Jenkins-PHP
jenkins-php.org

[Jenkins - ObjectStore](#)

- [Back to Dashboard](#)
- [Status](#)
- [Changes](#)
- [Workspace](#)
- [Build Now](#)
- [Delete Project](#)
- [Configure](#)
- [Checkstyle Warnings](#)
- [PMD Warnings](#)
- [Plots](#)
- [Clover HTML Report](#)
- [API Documentation](#)
- [Code Browser](#)
- [Violations](#)

Build History (trend)	
#31	22-Jul-2011 08:47:01
#30	22-Jul-2011 08:28:39
#26	21-Jul-2011 01:03:01
#25	21-Jul-2011 00:15:31
#24	21-Jul-2011 00:08:34
#23	18-Jul-2011 22:34:17
#22	18-Jul-2011 23:55:45
#21	18-Jul-2011 23:42:49
#20	18-Jul-2011 21:01:33
#19	18-Jul-2011 00:08:05
#18	18-Jul-2011 22:20:30
#17	18-Jul-2011 22:16:30

Project ObjectStore


[Clover HTML Report](#)

[API Documentation](#)

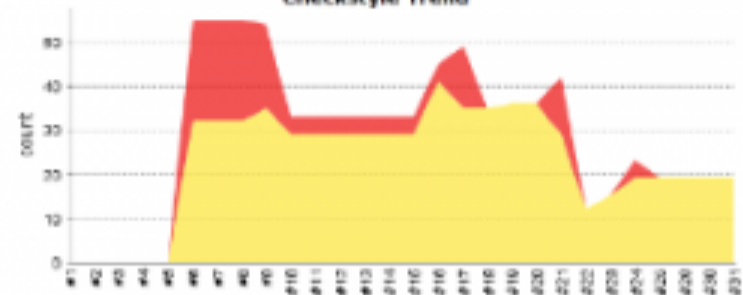
[Code Browser](#)

[Workspace](#)

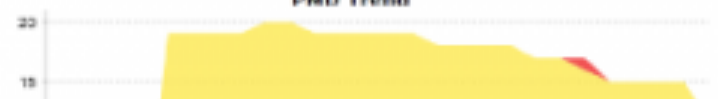
[Recent Changes](#)

[Latest Test Result \(14 failures / -4\)](#)
[Edit Description](#)
[Disable Project](#)

Checkstyle Trend


[Reloading Configuration](#)

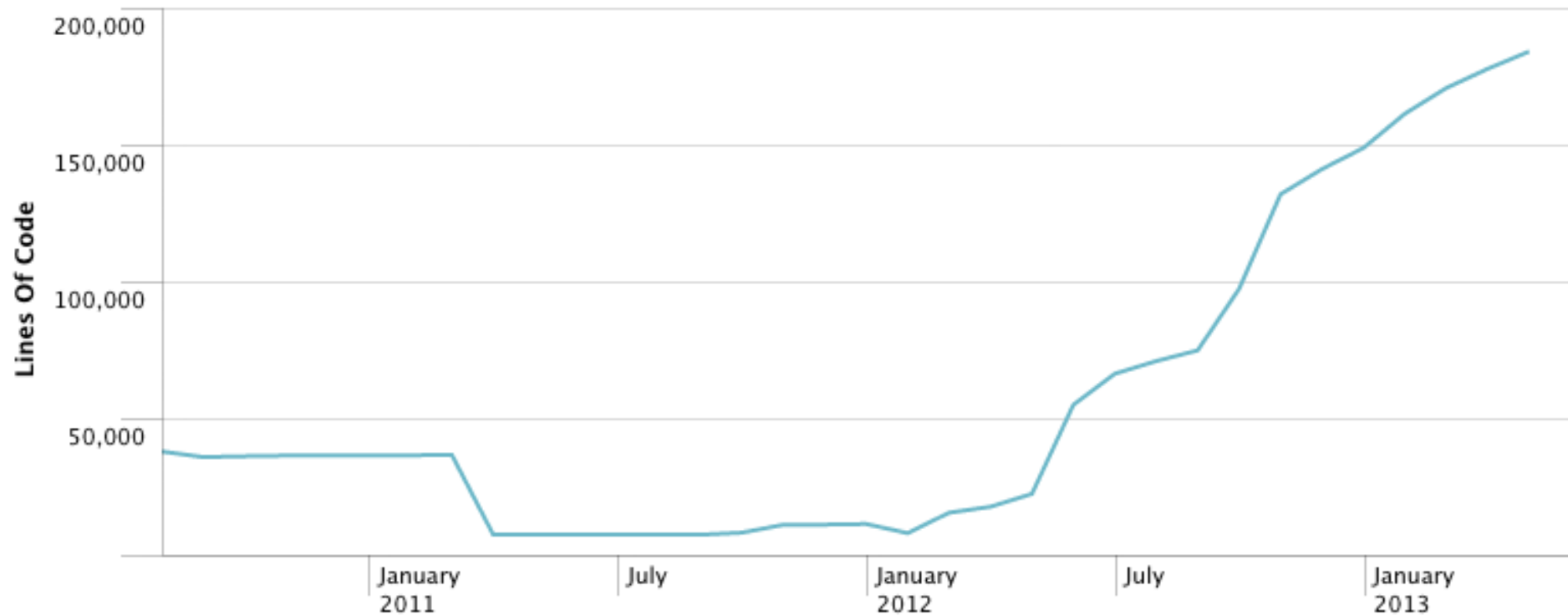
PMD Trend



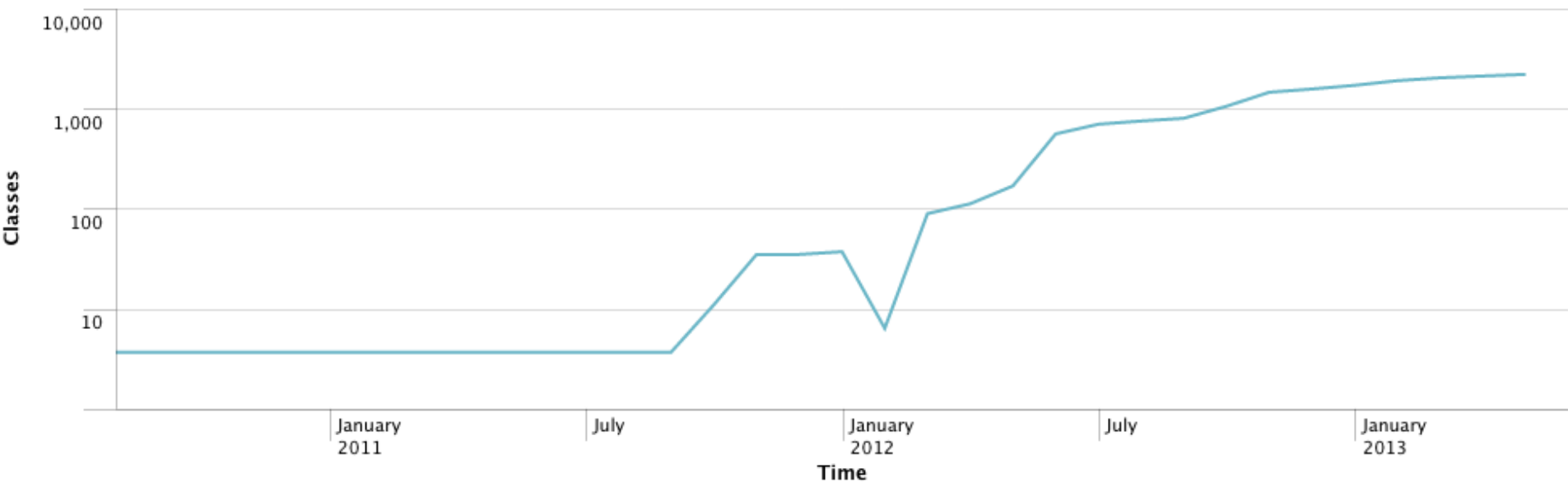
Sonar
SonarSource.Com

Time For Some Fun!

Drupal 8.x Branch Non-Comment Lines Of Code

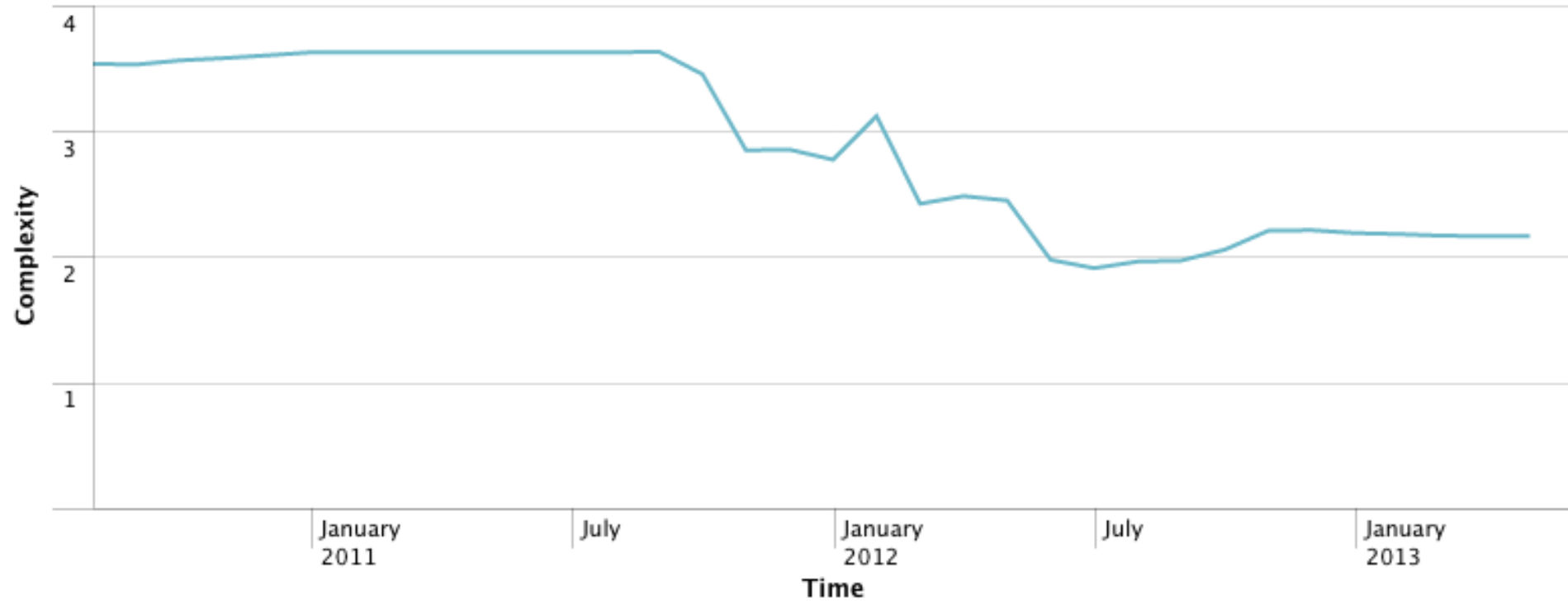


Drupal 8.x Branch Number Of Classes

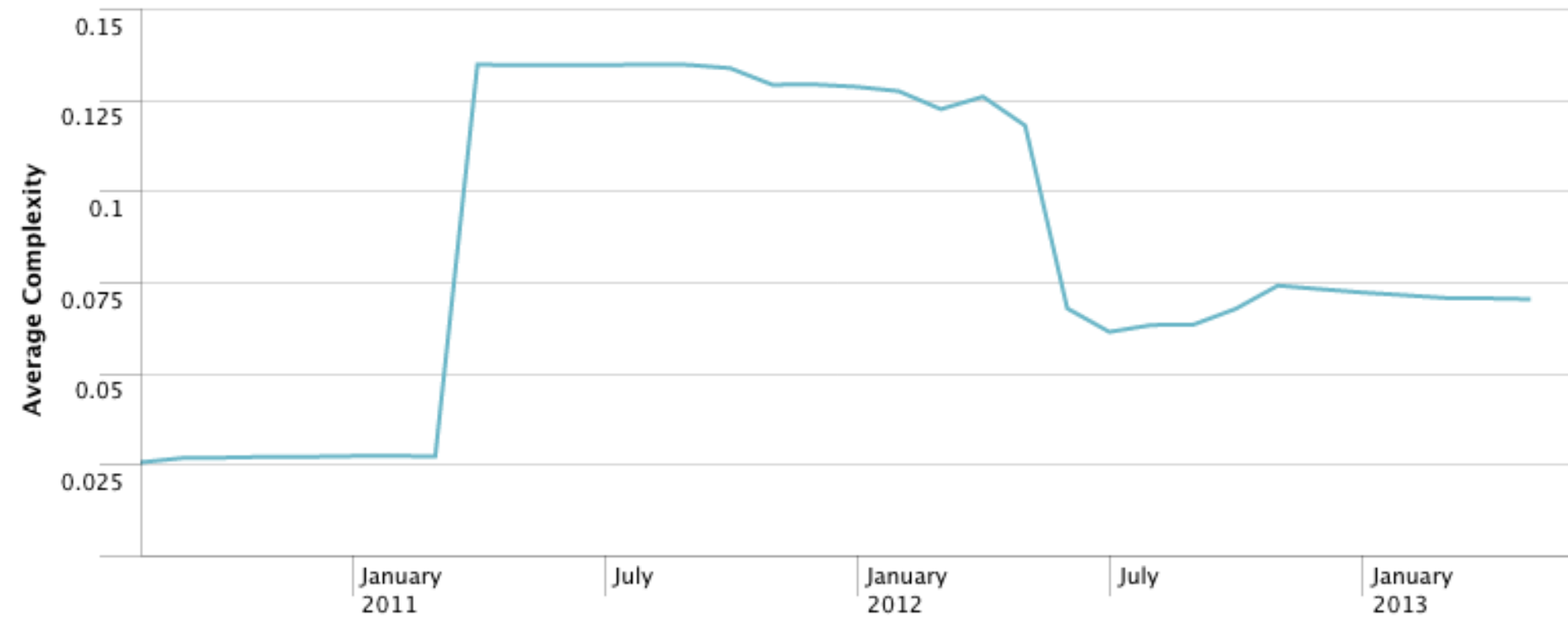


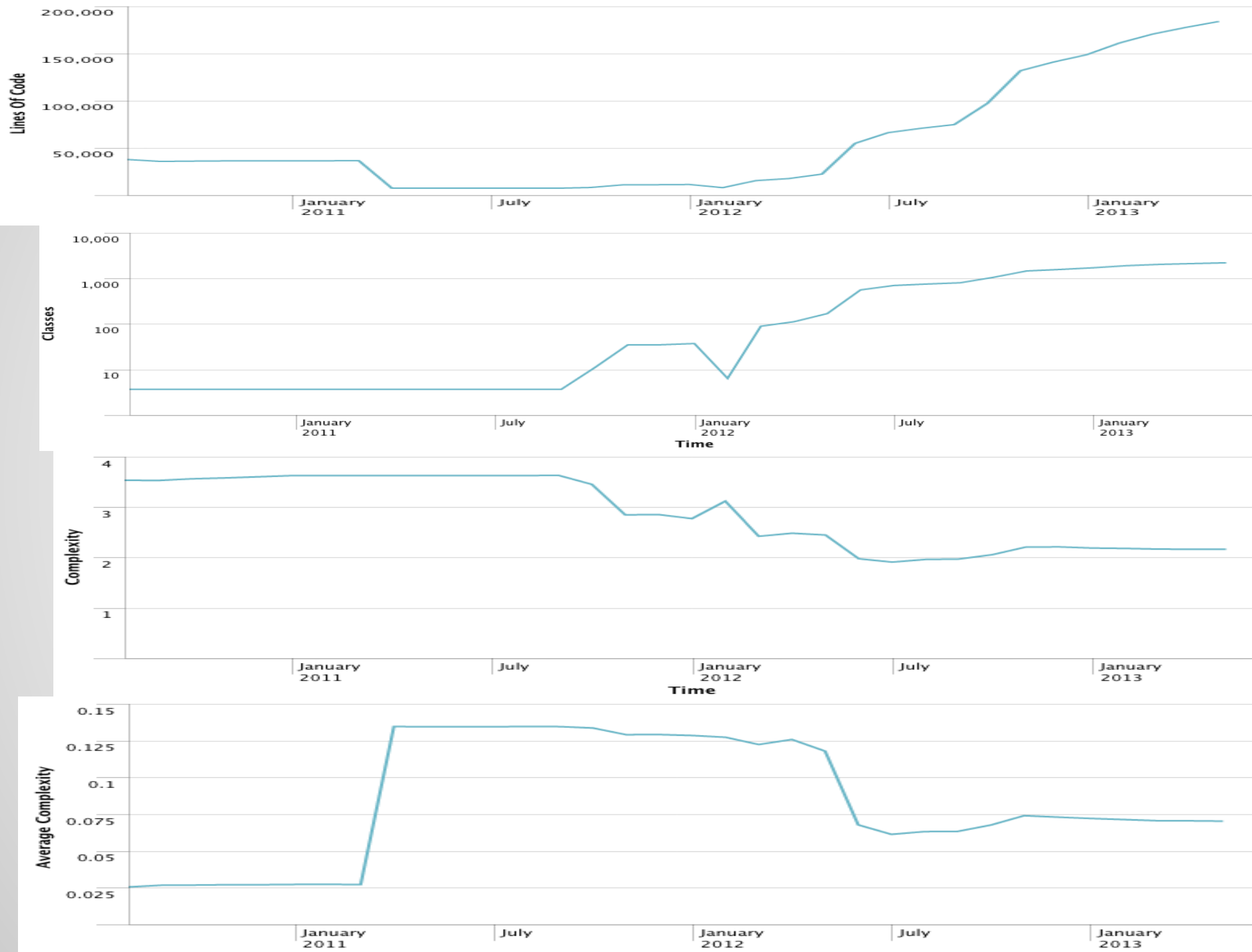
Drupal 8.x Branch

Cyclomatic Complexity Per Method



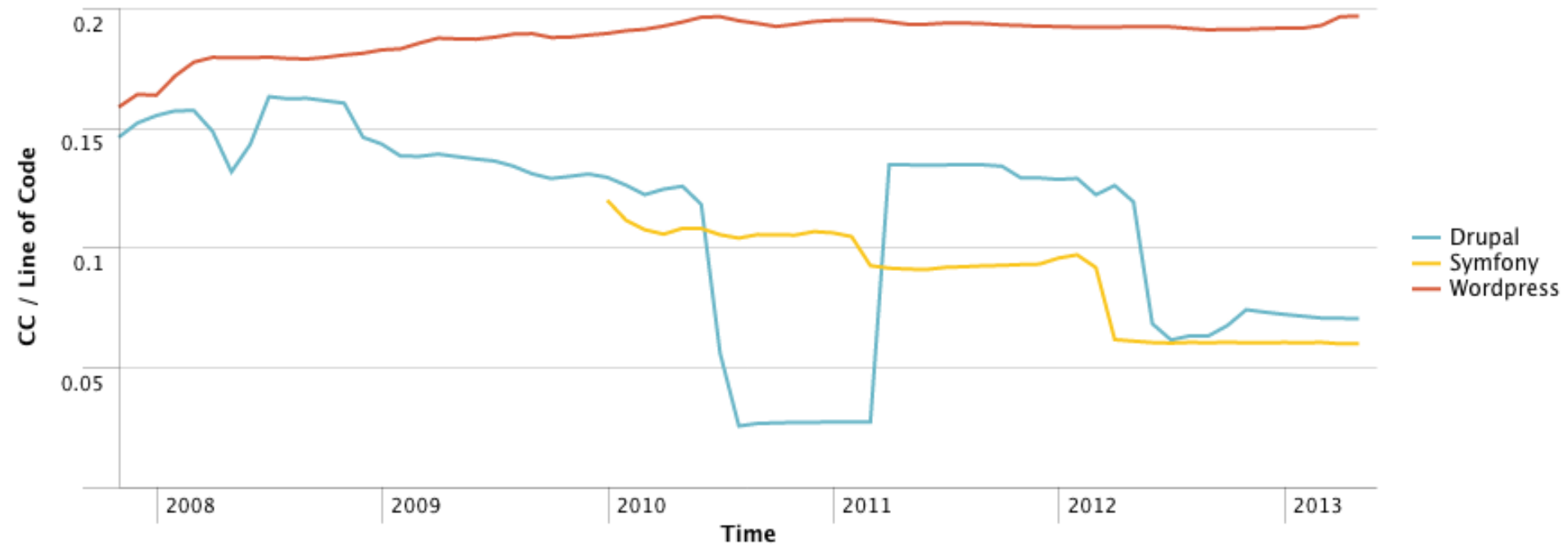
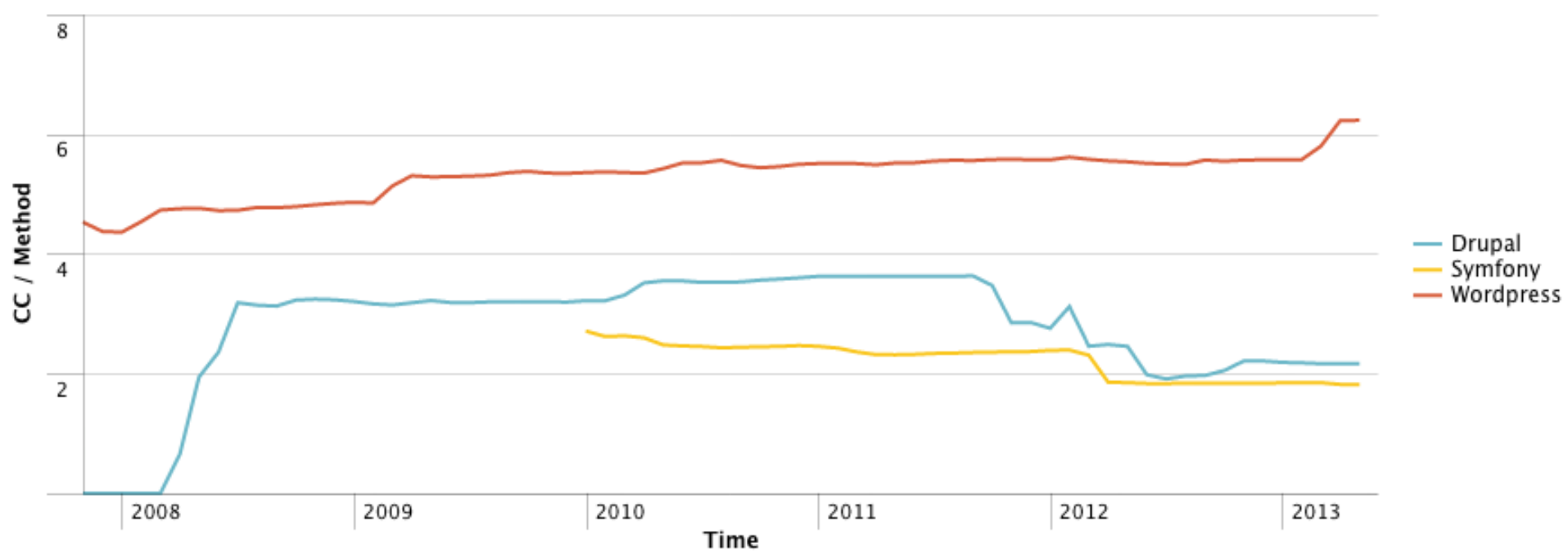
Drupal 8.x Branch Cyclomatic Complexity Per Line





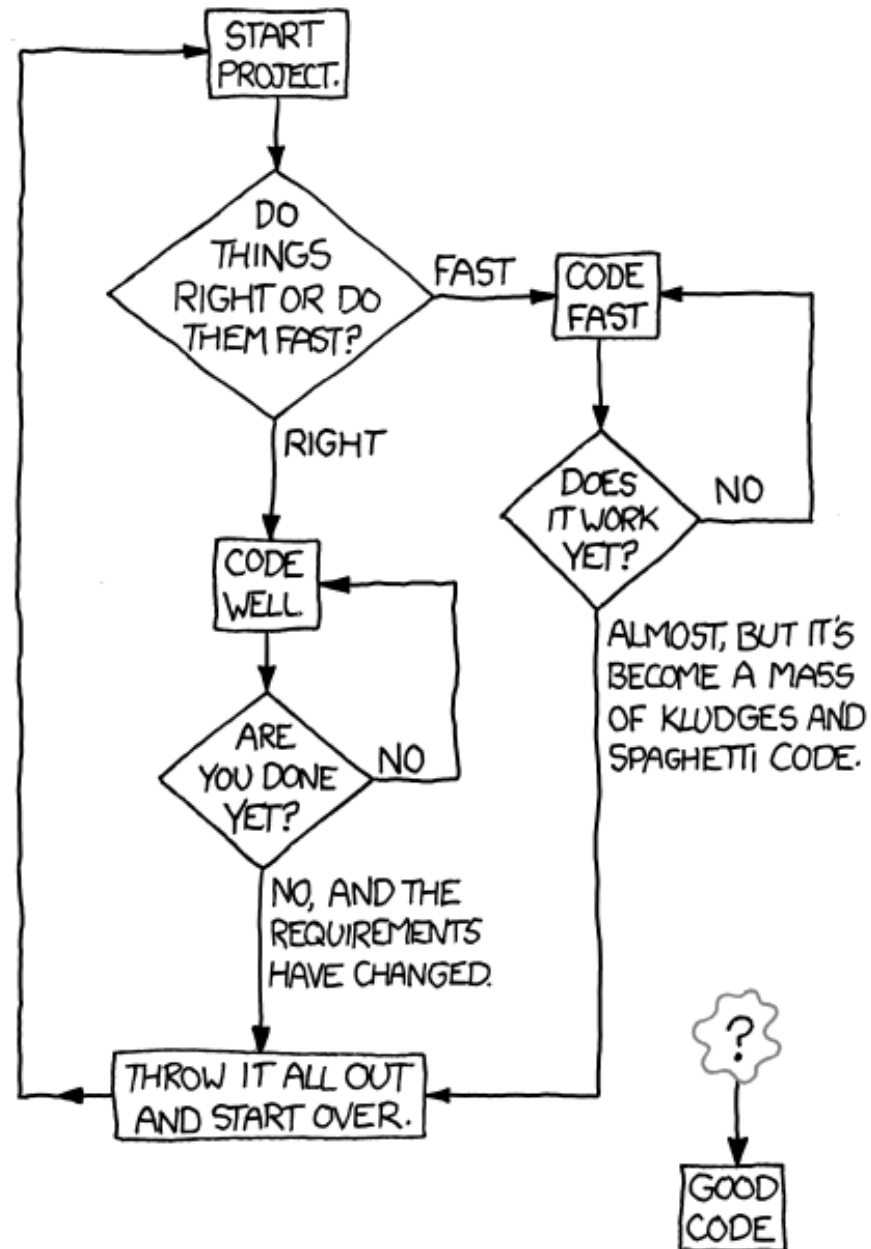
Success.





**One More Thing
To Keep In Mind**

HOW TO WRITE GOOD CODE:



ANY QUESTIONS ?



Anthony Ferrara

Joind.in/10513

@ircmaxell

me@ircmaxell.com

blog.ircmaxell.com

github.com/ircmaxell

youtube.com/ircmaxell