Evolution of Dynamic Feature Usage in PHP

Mark Hills

22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering (SANER 2015), ERA Track
March 2-4, 2015
Montreal, Canada

http://www.rascal-mpl.org

PHP Analysis in Rascal (PHP AiR)

- PHP AiR: a framework for PHP source code analysis
- Domains:
  - Program analysis (static/dynamic)
  - Software metrics
  - Empirical software engineering

PHP Analysis in Rascal (PHP AiR)

- PHP AiR: a framework for PHP source code analysis
- Domains:
  - Program analysis (static/dynamic)
  - Software metrics
  - Empirical software engineering

PHP Analysis in Rascal (PHP AiR)

- PHP AiR: a framework for PHP source code analysis
- Domains:
  - Program analysis (static/dynamic)
  - Software metrics
  - Empirical software engineering

What do we want? Soundness, precision…

- Example: static taint analysis
- Sound: we don’t want false negatives
    - We want to find all possible uses of “tainted” values in security-conscious code
- Precise: we don’t want false positives
    - We don’t want to report errors that are not *real* errors, i.e., that cannot cause problems at runtime

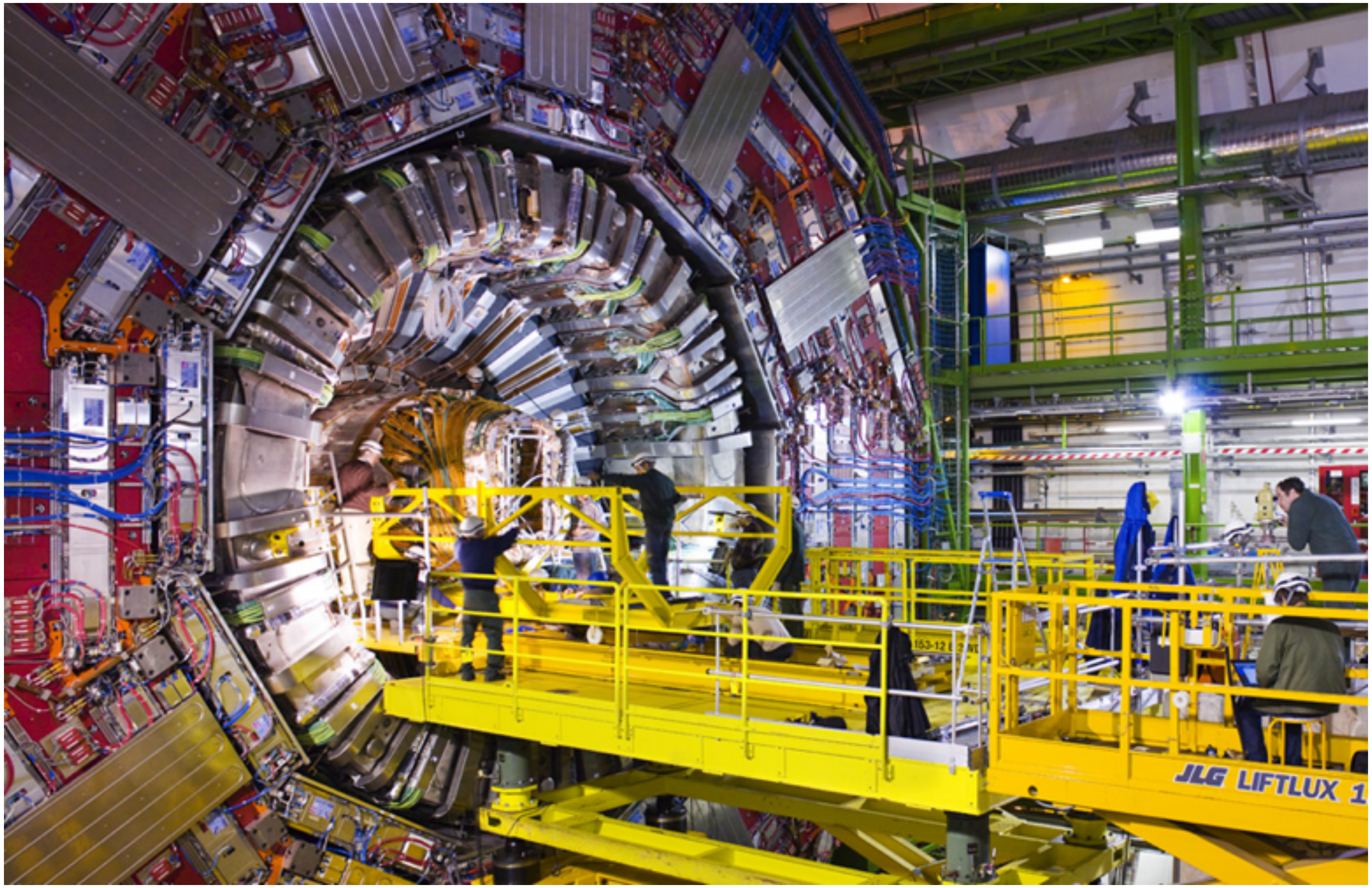
So, what’s the problem?

- Soundness and precision often conflict!
- We need to make engineering trade-offs to build realistic tools, make tools “soundy” and more precise
- We need to do this carefully, based on evidence:
  - Which features do we have to support?
  - Do we have to support dynamic features in their full generality?
  - Can we find patterns that we can exploit to help?

Here: determine usage patterns over time

- How has the profile of dynamic feature usage changed over the release history of PHP systems?
- Why has this changed? Why do we see features appear and/or disappear?
- Can we extract information (e.g., usage patterns) from this to help us build better program analysis tools?

Setting Up the Experiment: Tools & Methods



http://cache.boston.com/universal/site_graphics/blogs/bigpicture/lhc_08_01/lhc11.jpg

Building an open-source PHP corpus

- Original corpus: 19 open-source PHP systems, 3.37 million lines of PHP code, 19,816 files
- Select two systems: WordPress and MediaWiki
- Why these two?
    - Widely used, long release histories (2003 to now)
    - Study encompasses 93 releases of WordPress, 189 releases of MediaWiki, roughly 90 million SLOC

Methodology

- Scripted extract of releases from GitHub, all code parsed with an open-source PHP parser
- Dynamic features identified using pattern matching
- Raw numbers extracted to CSV files, trends computed with Rascal
- More in-depth explorations performed manually or using custom-written analysis routines
- All computation scripted, resulting figures and tables generated
    - http://www.rascal-mpl.org/

Which dynamic features?

- Variable Constructs
- Overloading
- eval

Which dynamic features?

- Variable Constructs
  - Lets you use variables instead of identifiers
  - Usable for variables, properties, class names, method and function names, etc.

```
$fields = array( 'views', 'edits', 'pages', 'articles', 'users', 'images' );
foreach ( $fields as $field ) {
    if ( isset( $deltas[$field] ) && $deltas[$field] ) {
        $update->$field = $deltas[$field];
    }
}
```

Which dynamic features?

- Overloading
  - Handles access to undefined or non-visible properties and methods

```
function __call( $fname, $args ) {
    $realFunction = array( 'Linker', $fname );
    if ( is_callable( $realFunction ) ) {
        wfDeprecated( get_class( $this ) . '::' . $fname, '1.21' );
        return call_user_func_array( $realFunction, $args );
    } else {
        $className = get_class( $this );
        throw new MWException( “…” );
    }
}
```

Which dynamic features?

- eval
  - evaluates arbitrary PHP code

```
while ( ( $line = Maintenance::readconsole() ) !== false ) {
    // elided...
    try {
        $val = eval( $line . ";" );
    } catch ( Exception $e ) {
        echo "Caught exception " . …
        continue;
    }
    // elided...
}
```

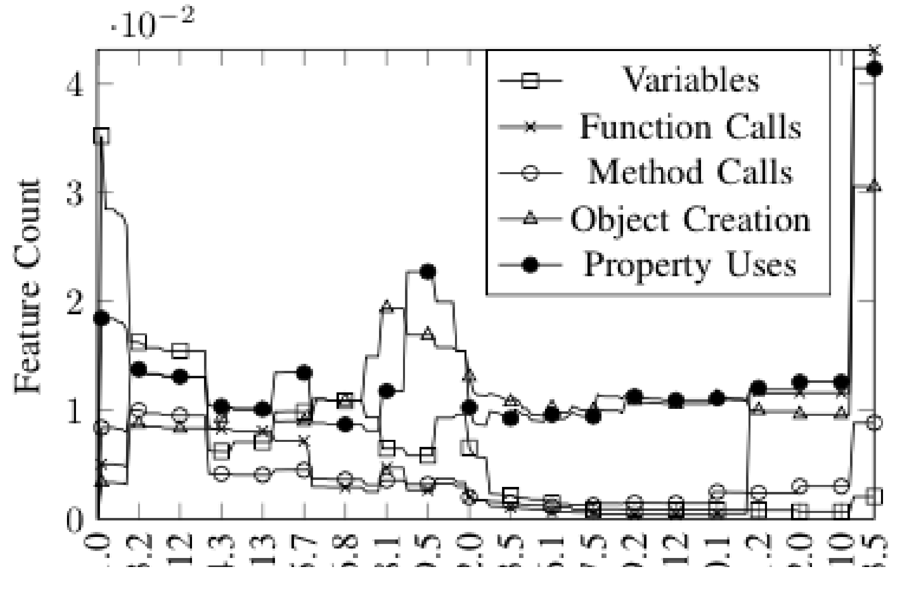
Threats to validity

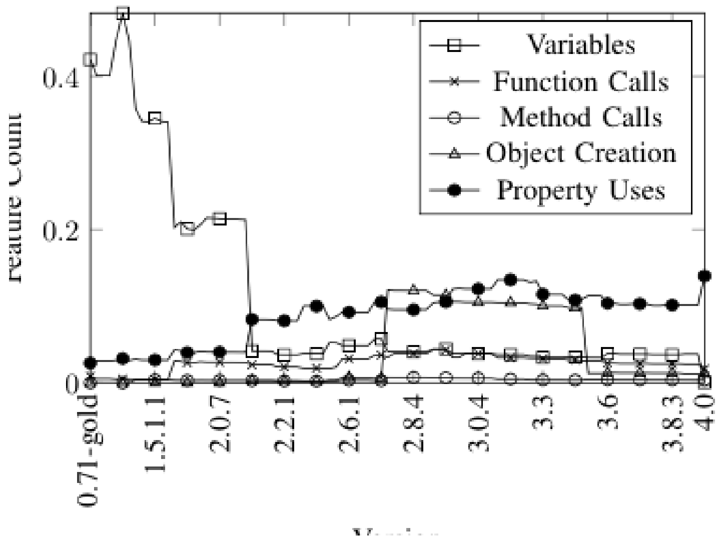
- Results could be very specific to either WordPress or MediaWiki

Threats to validity

- Results could be very specific to either WordPress or MediaWiki
- Mitigation: expanding to include other systems, plus results seem reasonable based on earlier work

Interpreting the Results





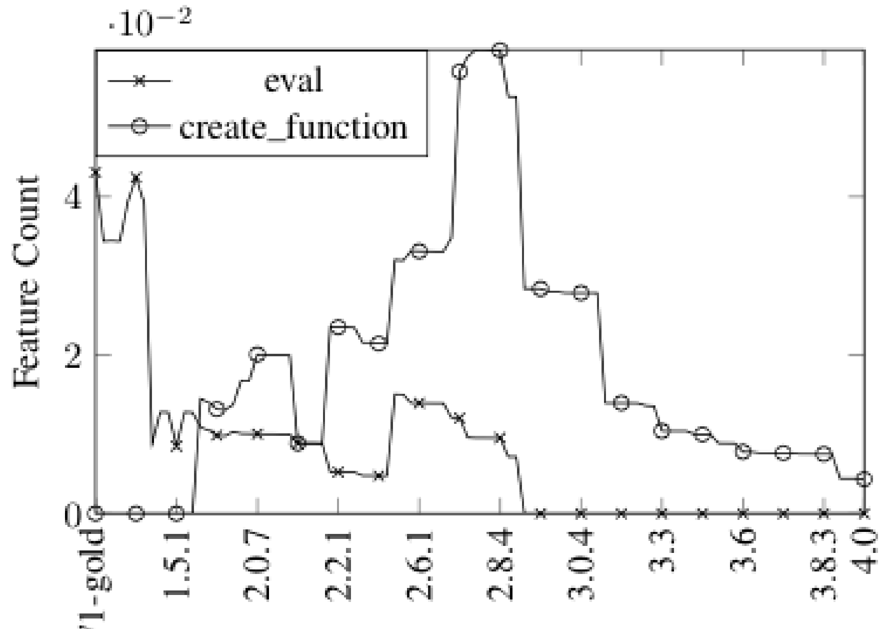


Fig.

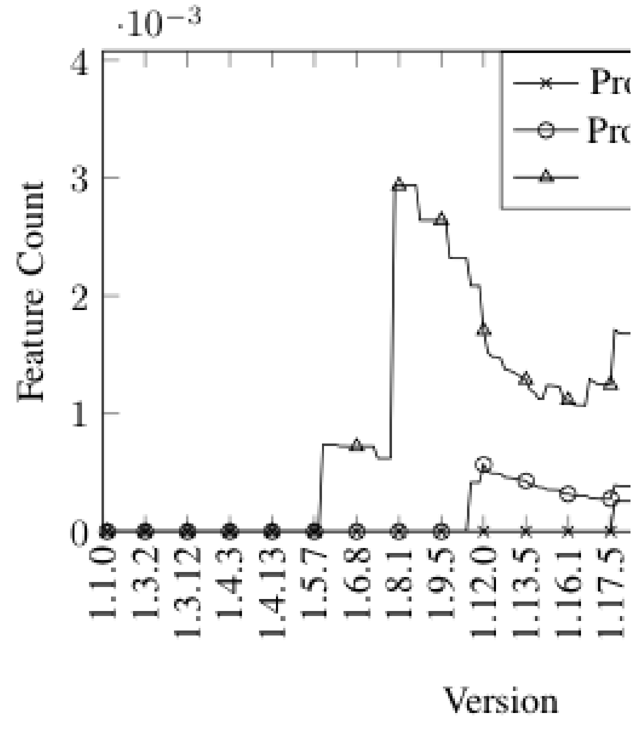


Fig. 5. Magic Methods in MediaWiki, Scaled by

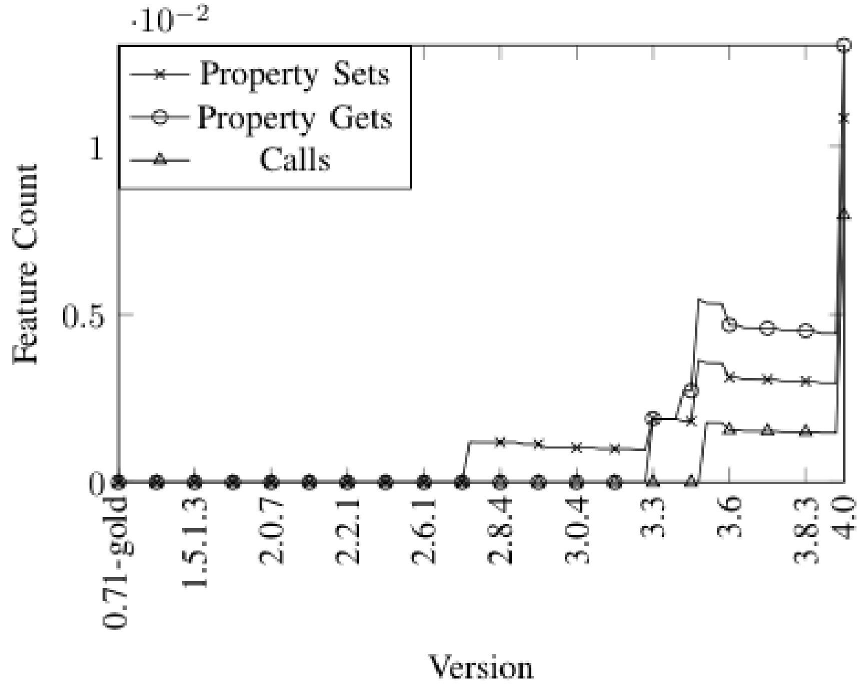


Fig. 4. Magic Methods in WordPress, Scaled by SLOC.

TABLE I. THE CORPUS, FIRST AND LAST RELEASES.

System	Version	PHP	Released	File Count	SLOC
WordPress	0.71	4.0.6	June 9, 2003	54	11,613
	4.0	5.2.4	September 4, 2014	481	138,414
MediaWiki	1.1.0	4.3.2	December 8, 2003	146	45,230
	1.23.5	5.3.2	October 1, 2014	1,629	294,731

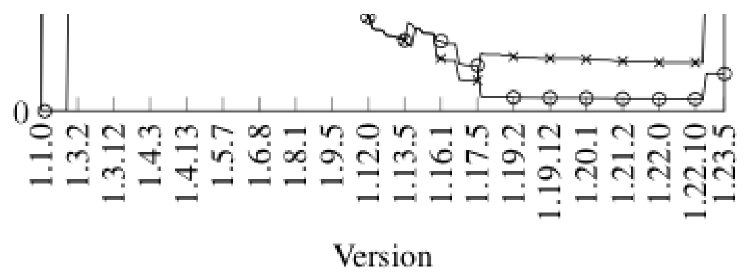


Fig. 7. Eval Constructs in MediaWiki, Scaled by SLOC.

Zooming in: Variable Features

- Variable properties are becoming more common (why? speculation: PHP is now OO, more code is moving to use OO features)
- Variable variables common in some systems, decreasing in others
- Differences in usage between different applications = no overall trend for many of these features
- There may be patterns we can exploit here for better precision…

A pattern example…

```
$fields = array( 'views', 'edits', 'pages', 'articles', 'users', 'images' );
foreach ( $fields as $field ) {
    if ( isset( $deltas[$field] ) && $deltas[$field] ) {
        $update->$field = $deltas[$field];
    }
}
```

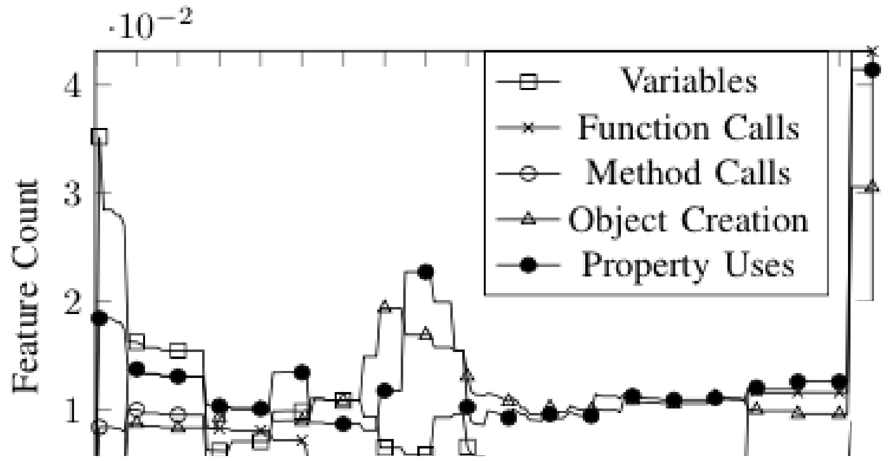
Zooming in: Overloading

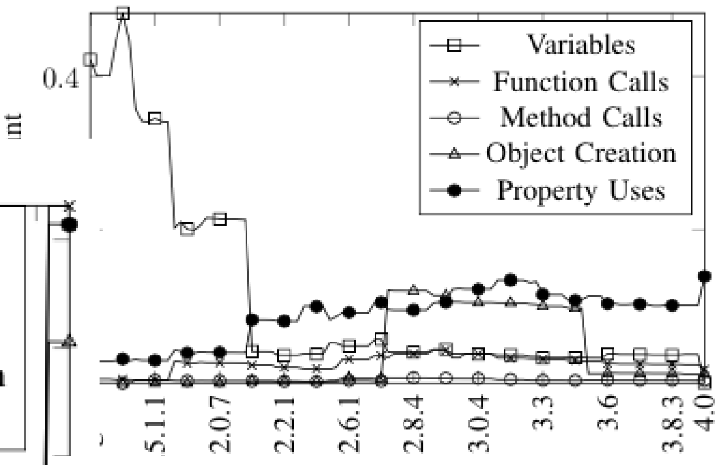
- Fairly stable in MediaWiki, with a spike at the end caused by a decrease in SLOC
- Increasing use in WordPress
- Still rare, but becoming more important
- Need type inference to really know impact: how often are these actually used? (we’re working on this now…)

Zooming in: eval and create_function

- Never popular, trend moving generally down
- Many uses replaced with callbacks (still dynamic, but *less* dynamic)
- Remaining uses in MediaWiki for admin, testing
- Libraries are important here too: PCLZip in WordPress was the source of most of the eval uses there…

Summary





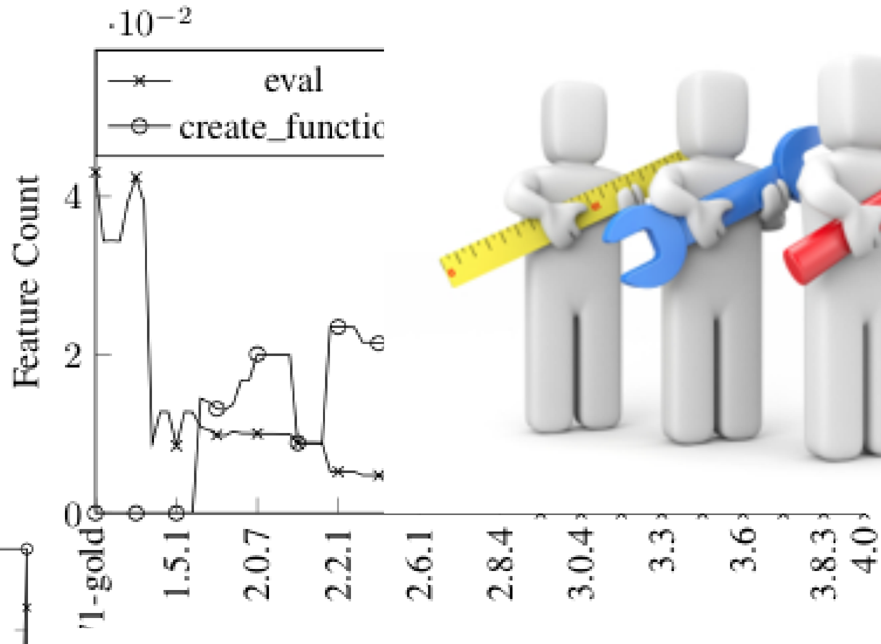


Fig.

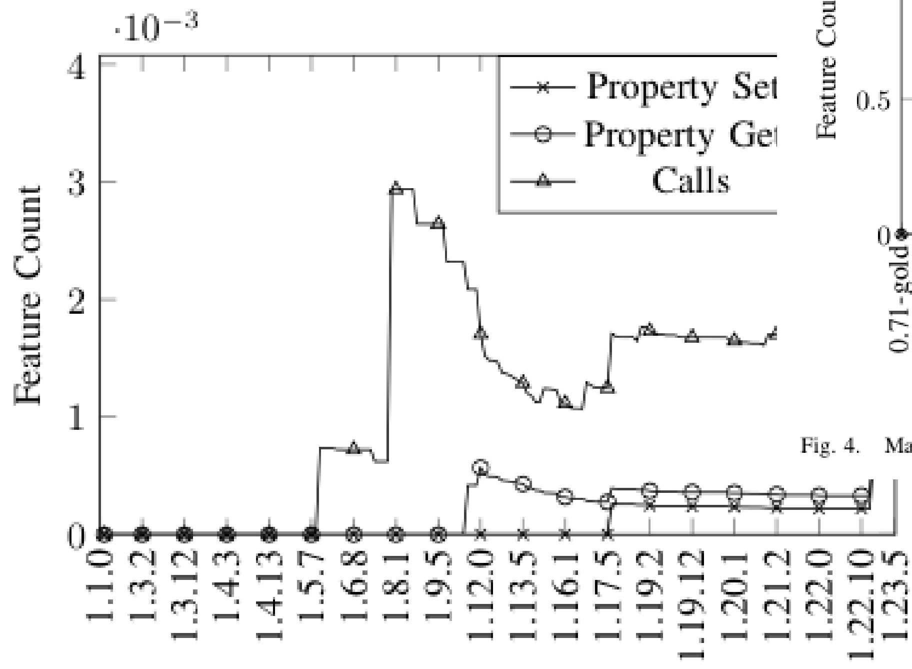


Fig. 5. Magic Methods in MediaWiki, Scaled by SLOC.

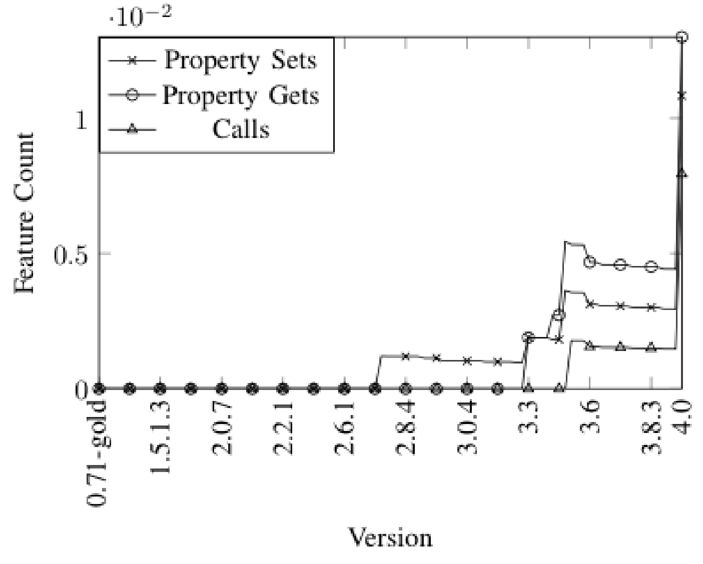
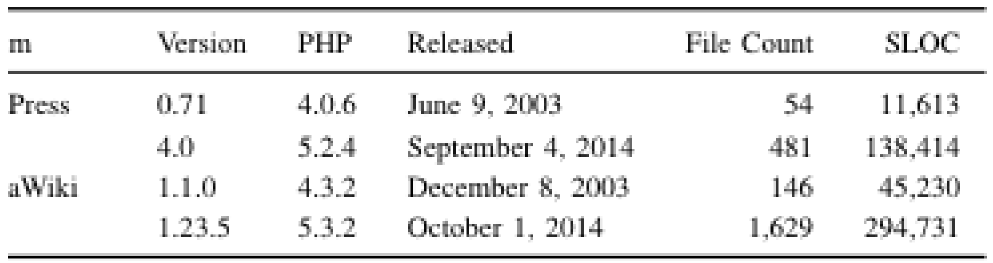


Fig. 4. Magic Methods in WordPress, Scaled by SLOC.

TABLE I. THE CORPUS, FIRST AND LAST RELEASES.



m	Version	PHP	Released	File Count	SLOC
Press	0.71	4.0.6	June 9, 2003	54	11,613
	4.0	5.2.4	September 4, 2014	481	138,414
aWiki	1.1.0	4.3.2	December 8, 2003	146	45,230
	1.23.5	5.3.2	October 1, 2014	1,629	294,731

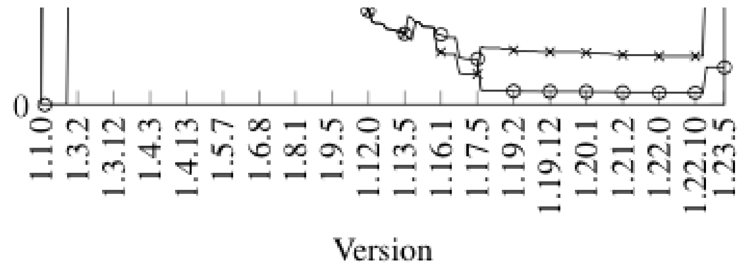
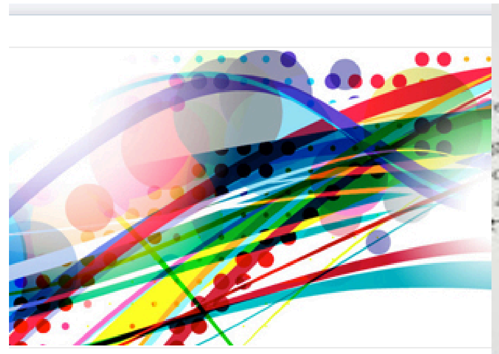
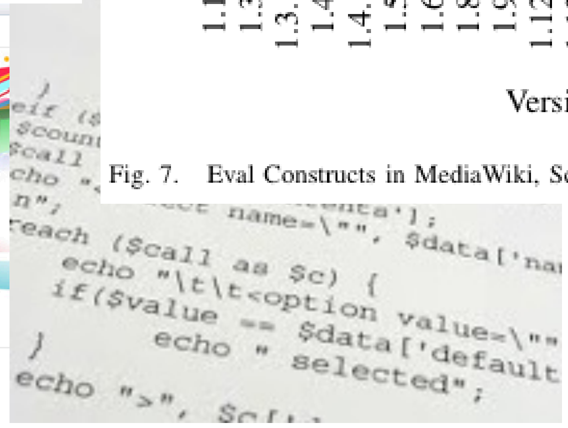


Fig. 7. Eval Constructs in MediaWiki, Scaled by SLOC.



4 | By: Patrick Mulvey | Posted: July 12, 2013 | News & Opinion

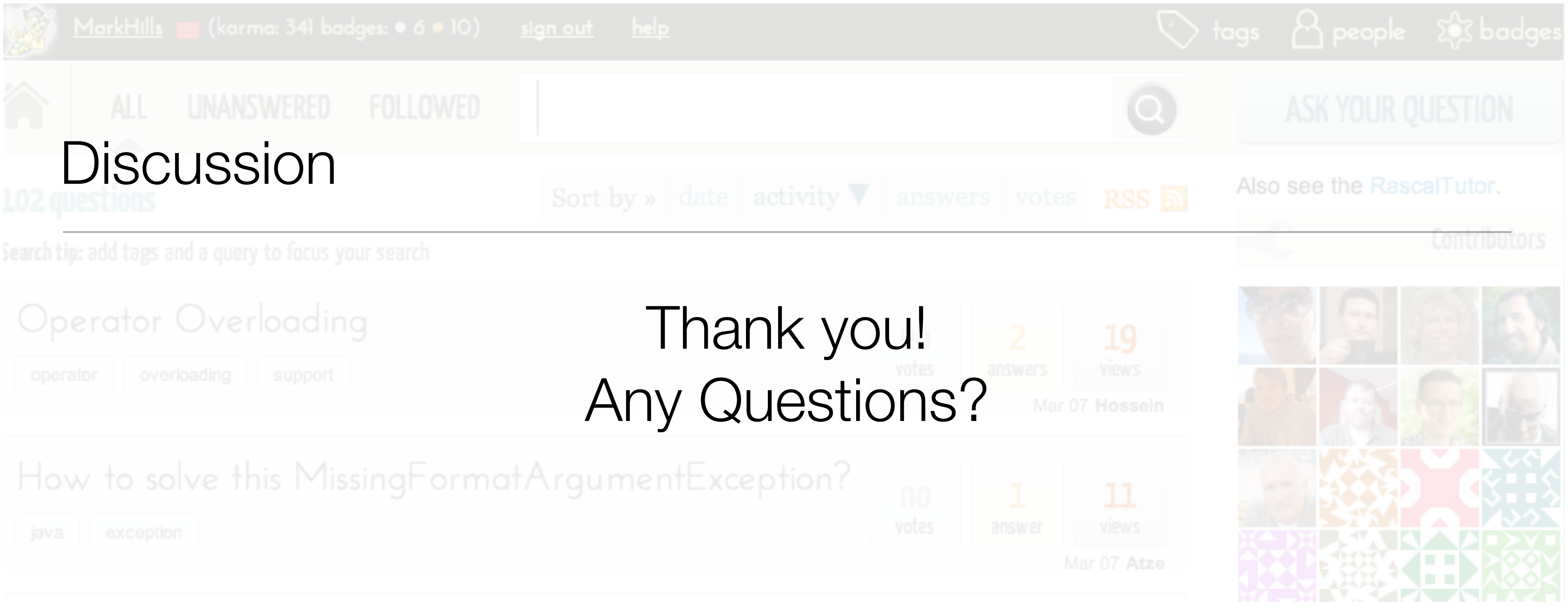


```
eif ($
$count
$call
cho ",
n";
      ect name=\"", $data['na
reach ($call as $c) {
  echo "\t\t<option value=\""
  if($value == $data['default
     echo " selected";
  }
echo ">", $c[
```

What have we learned? What’s left?

- Variable features need to be modeled, variable properties are becoming more common, patterns may help
- Overloads are still rare, but we need ways to detect where they are used
- Eval and create_function are, thankfully, quite rare
- Future: need to expand the feature set and corpus
  - Non-covered variants, other dynamic features
  - Cover more systems, further expand corpus

Discussion


MarkHills (karma: 341 badges: ● 6 ● 10) sign out help

tags people badges

ALL UNANSWERED FOLLOWED

ASK YOUR QUESTION

102 questions

Sort by » date activity ▼ answers votes RSS

Search tip: add tags and a query to focus your search

Also see the RascalTutor.

Contributors

Operator Overloading

operator overloading support

2 answers 19 views

Mar 07 Hossein

How to solve this MissingFormatArgumentException?

java exception

no votes 1 answer 11 views

Mar 07 Atze

Thank you!

Any Questions?

- Rascal: http://www.rascal-mpl.org
- Me: http://www.cs.ecu.edu/hillsma