East Carolina UNIVERSITY

Research in Meta-Programming, Program Analysis, and Software Analytics

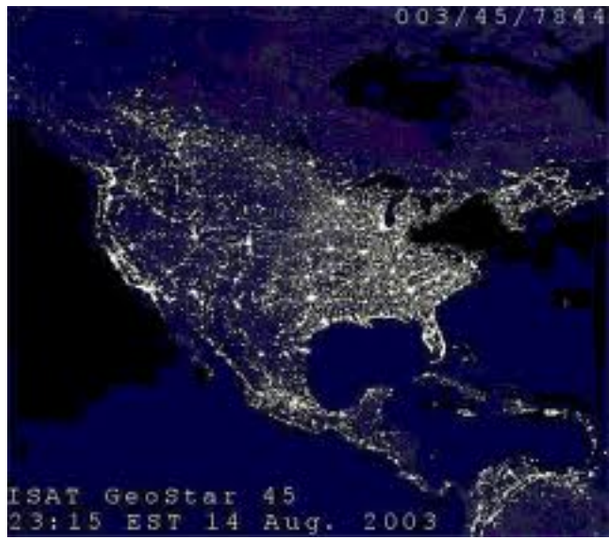
Prof. Mark Hills
Department of Computer Science
East Carolina University
May 25, 2016

http://www.rascal-mpl.org

Bugs are chaotic, expensive, & potentially fatal

- November 2012: Swedish Stock Exchange disrupted when an order worth over £43 trillion is placed due to a parsing error

- August 2012: Knight Capital loses $440 million in 45 minutes because of a bug in trading software



- August 2003: A software bug in alarm software causes the Northeast Blackout in the US and Canada, impacting 55 million people

- 1985 to 1987: A concurrency bug in the Therac-25 gives massive overdoses of radiation in at least six accidents; three patients later died

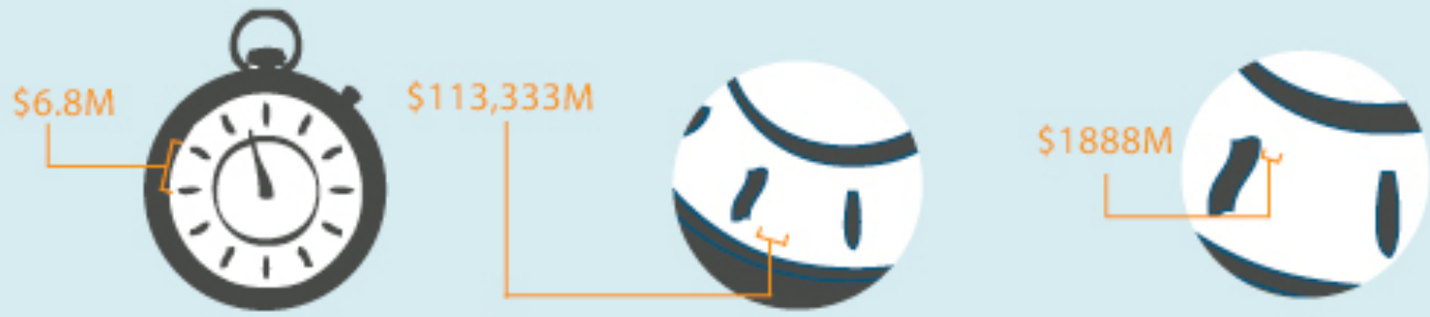
The Severity of Bugs: Are We Doomed?

Software defects cost the US economy alone over **$60 billion** yearly

This is over $6.8 million per hour

Which is over $113,333 per minute

And $1,888 per second



This is as much money as the US spent on airport security in the last decade, and more valuable than Rupert Murdoch's News Corp

Source: NIST

The cost of correcting bugs during the testing (QA) stage

$7,136

The cost of correcting bugs after release is

$14,102

The cost of correcting bugs during coding is

$937

Source: B. Boehm and V. Basil, Software Defect Reduction Top 10 List, IEEE Computer, January 2001

http://www.codeguru.com/blog/category/programming/the-cost-of-bugs.html

Program Analysis to the Rescue!

Static program analysis is the analysis of computer software that is performed without actually executing programs (analysis performed on executing programs is known as dynamic analysis). In most cases the analysis is performed on some version of the source code and in the other cases some form of the object code. The term is usually applied to the analysis performed by an automated tool, with human analysis being called program understanding, program comprehension or code review.

- http://en.wikipedia.org/wiki/Static_program_analysis

Why Static Analysis?

- Automated

- Reasons over all possible executions, don’t need to worry about missing execution paths

- Lays foundation for powerful tool support: refactoring, code comprehension, type inference, bug finding

- Can give programmers checkable ways to make implicit information explicit: numeric ranges, nullity, units of measurement

Why Dynamic Analysis?

- More precise, reasons over actual executions
- Potentially more suited to dynamic languages: dynamic features challenging to analyze
- Can work in tandem with static analysis: dynamic analysis results can make static analysis more precise
- Generally easier to implement

Wish List: what should we use to build analysis tools?

- First-class support for working with concrete and abstract program representations
- Code that is close to algorithms
  - data structures like sets, relations, tuples, and lists
  - pattern matching & fixpoint computation
- IDE support, for interaction with developers
- Support for computation needed in empirical research

Rascal: The Meta-Programming One Stop Shop

- “Rascal is a domain specific language for source code analysis and manipulation a.k.a. meta-programming.” (http://www.rascal-mpl.org/)

- Language focus: program analysis, program transformation, domain-specific language creation

- Current projects across large numbers of domains, both within and outside academia

- Open source, committers worldwide

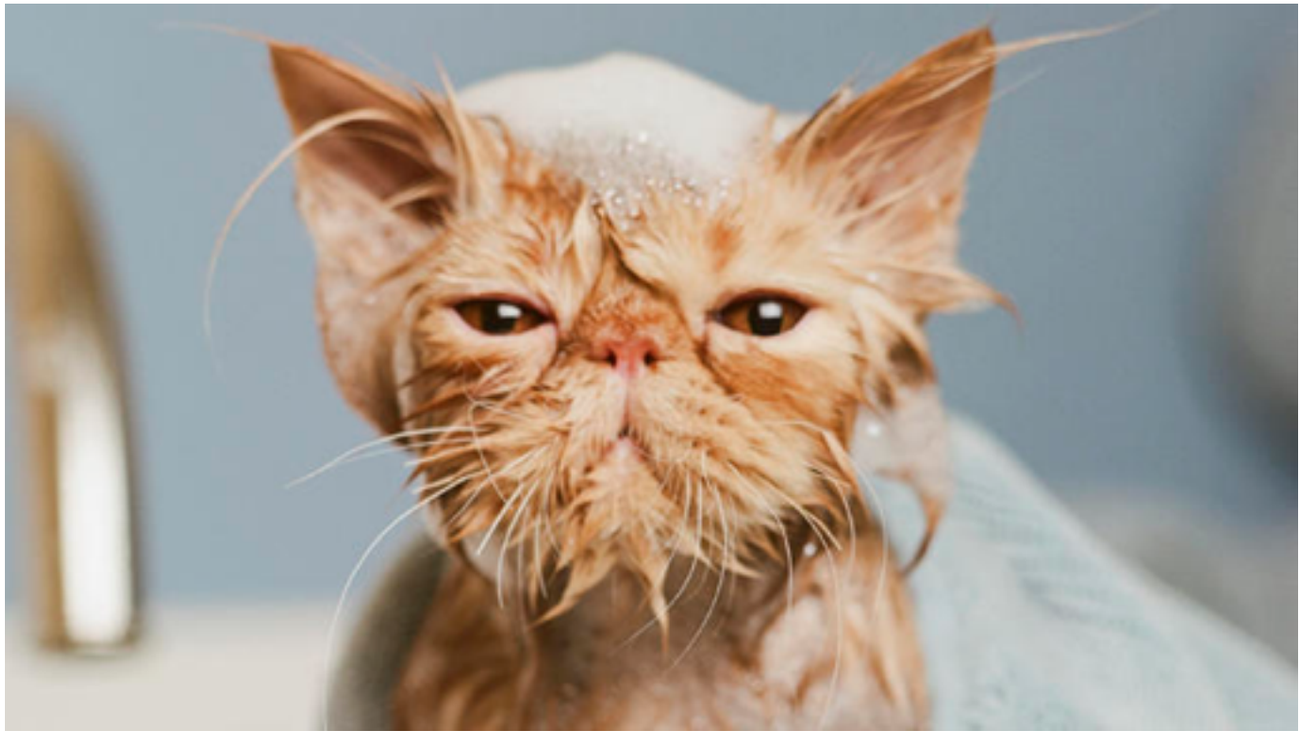
Rascal Benefits (Not an Exhaustive List!)

- Built-in language support for matching & transforming code
- Rich data types: relations, maps, lists, sets, tuples, parse trees, higher-order functions
- Console supports interactive exploration
- Extensible with Java and Eclipse
- Empirical research support: code querying, statistical analysis, interaction with external data (e.g., code repositories, external databases), visualization

My research: what am I trying to do?

- Big picture: develop a framework for PHP analysis
- Specifics:
  - Empirical software engineering
  - Software metrics
  - Program analysis (static/dynamic)
  - Developer tool support

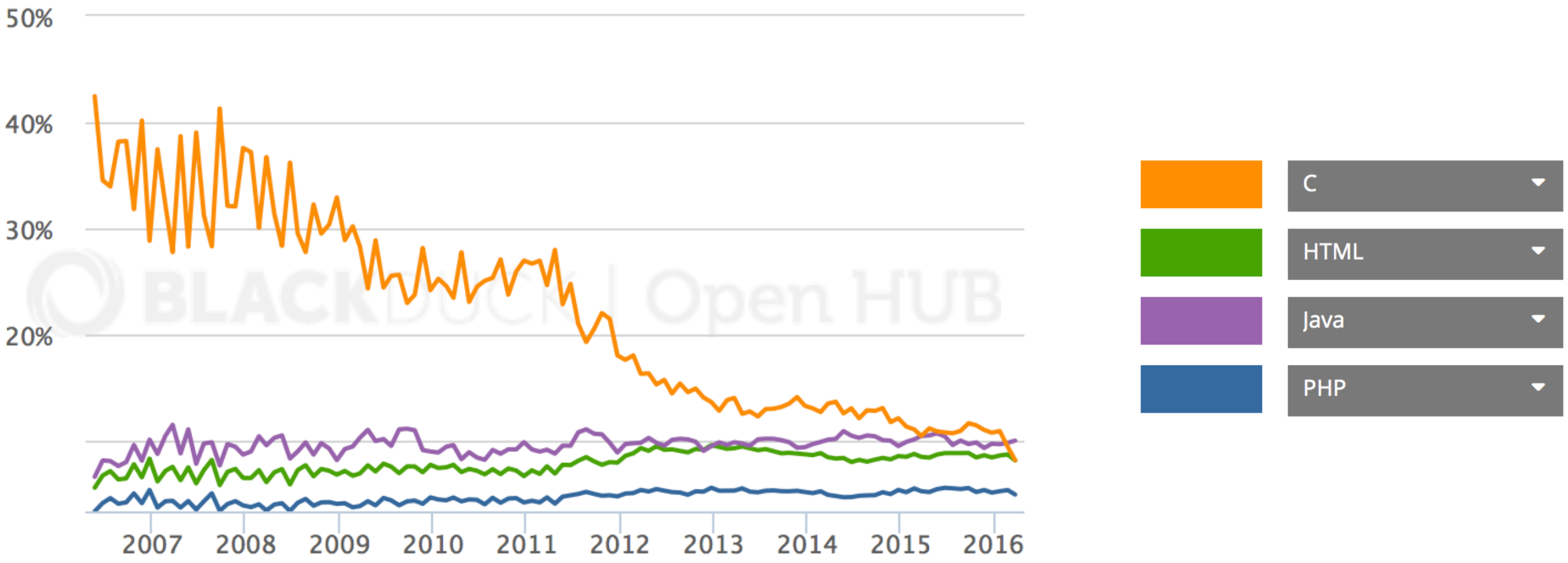
Why look at PHP applications?



PHP applications are everywhere!

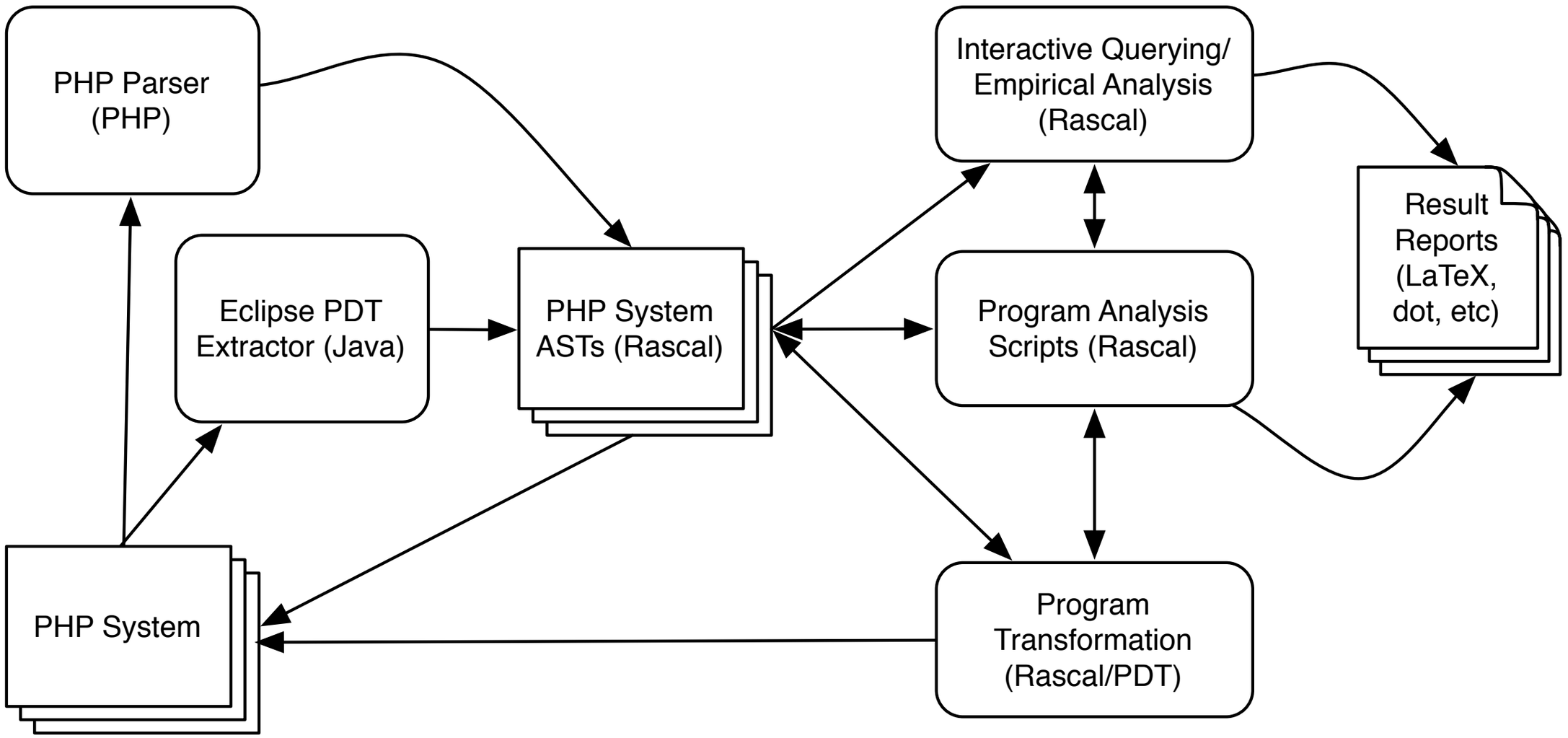
The New York Times

Open Source Commits by Language (Ohloh.net)



https://www.openhub.net/languages/compare?measure=commits&percent=true

Result: PHP AiR (Analysis in Rascal)



PHP AiR Design Points

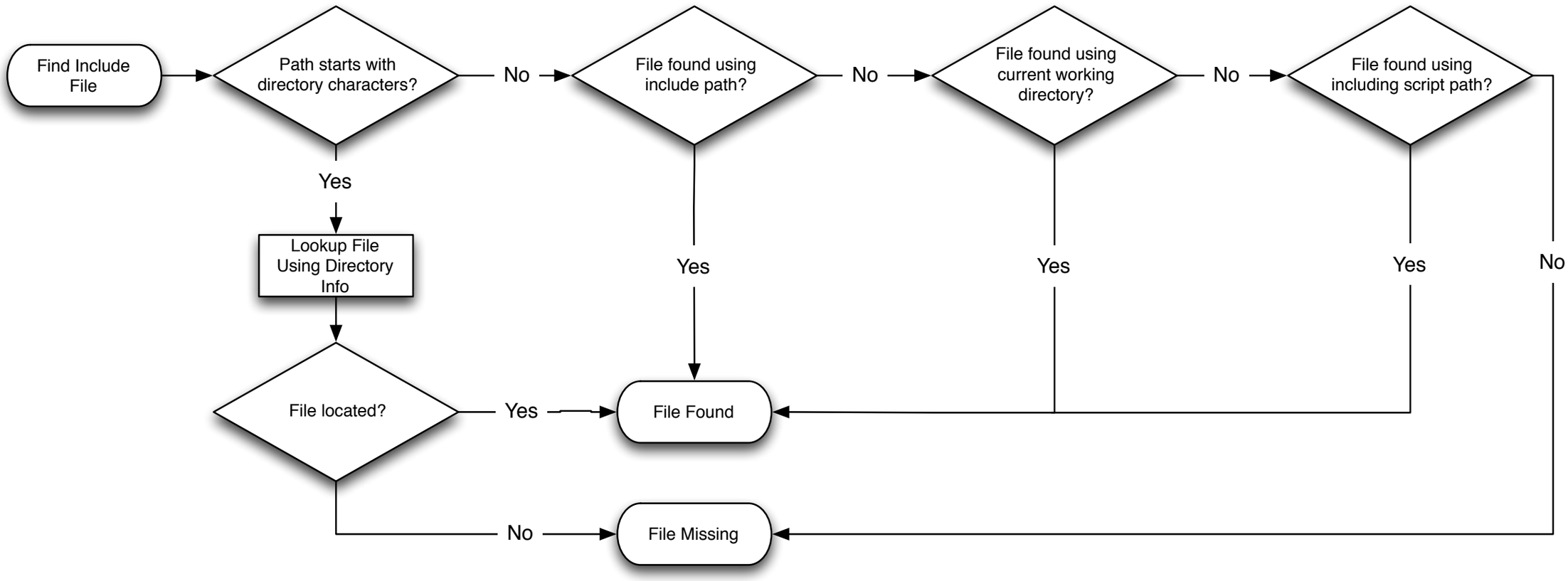
- Support for multiple PHP parsers
- Want to support integration with Eclipse (part way there)
- Perform tasks by writing Rascal code (not focused on push-button solutions, goes with Rascal “no magic” principle)
- Want to work with real PHP code (WordPress, MediaWiki, etc)
- See “Enabling PHP Software Engineering Research in Rascal”, by Hills, Klint, and Vinju, for more: http://dx.doi.org/10.1016/j.scico.2016.05.003

Example: Empirical Study of PHP Feature Usage

- Perspective: Creators of program analysis tools
- What does a typical PHP program look like?
- What features of PHP do people really use?
- How often are dynamic features, which are hard for static analysis to handle, used in real programs?
- When dynamic features appear, are they really dynamic? Or are they used in static ways?

"An Empirical Study of PHP Feature Usage: A Static Analysis Perspective", Hills, Klint, and Vinju, ISSTA 2013.

Example: Resolving PHP Includes



- Resolving PHP includes is messy (see above)
- How many can we resolve at a per-file level? Per-program?

“Static, Lightweight Includes Resolution for PHP”, Hills, Klint, and Vinju, ASE 2014.

Example: Resolving PHP Variable Features

- Variable features in PHP defer selection of an identifier until runtime

- Patterns in the code can help us to identify the identifiers that will be used at runtime, *if* we can detect them

```php
// MediaWiki, /includes/Sanitizer.php, lines 424-428
$vars = array( 'htmlpairsStatic', 'htmlsingle',
  'htmlsingleonly', 'htmlnest',
  'tabletags', 'htmllist', 'listtags',
  'htmlsingleallowed', 'htmlelementsStatic' );
foreach ( $vars as $var ) {
  $$var = array_flip( $$var );
}
```


“Variable Feature Usage Patterns in PHP”, Hills, ASE 2015.

Example: Understanding WordPress Plugins

- How do developers use WordPress plugin features?
- How can we help developers to find the right extension points?
- Text mining and search can find the right extension points; static analysis over plugin *summaries* can let us link to the right handlers

```
// WordPress 4.2.4, wp-includes/meta.php, line 480
apply_filters( "get_{$meta_type}_metadata", null, $object_id, $meta_key, $single )

// Responsive Nagivation plugin, metabox/helpers/cmb_Meta_Box_Ajax.php, line 112
add_filter( 'get_post_metadata', array( 'cmb_Meta_Box_ajax', 'hijack_oembed_cache_get' ), 10, 3 )
```


"Navigating the WordPress Plugin Landscape", Hills, ICPC 2016.

Project 1: Is JavaScript still eval?

- In 2010 and 2011, Richards and co-authors published two papers about the behavior of JavaScript
  - First paper focused on uses of features challenging for static analysis to handle
  - Second focused specifically on the eval construct, looking at how it was used and if other language features could be used instead

Image borrowed from http://plg.uwaterloo.ca/~dynjs/

Project 1: Is JavaScript still eval?

- REU project goals
  - Have these numbers changed?
  - If so, why?

- Many changes to how JavaScript is used over the past few years, how does this effect the prevalence of features like eval?

- Project is combination of empirical analysis, dynamic analysis, and tool development (hopefully existing tools can be used for most of analysis)

Image borrowed from http://plg.uwaterloo.ca/~dynjs/

Project 2: Evolution of RTP Open-Source Activity

THE RESEARCH TRIANGLE PARK

- Lindsey Lanier, an ECU MSSE student, investigated open-source activity in RTP area

- Investigation used GHTorrent, which mirrors all data from GitHub in an easily-queried format (see http://ghtorrent.org/ for details)

- REU project goals
  - Add a temporal aspect to analysis: how has this changed over time?
  - Ensure this can be generalized to other regions

Project 3: Feature Usage in PHP

- ISSTA 2013 paper explored which features of PHP are used in actual programs, how often dynamic features are used, what features are needed to form an analyzable core of the language
- REU project goals
  - Revisit and expand upon this work — what has changed over the years, and why (if we can figure that out!)
  - Perform more detailed analyses in some cases (e.g., feature counts) where the original results were not as useful as they could be, results should be useful for tool builders

Project 4: Feature Evolution in PHP

- SANER 2015 paper started to look at this question
  - How has use of dynamic features changed over time?
- Goals of REU project:
  - Expand this to other systems
  - Expand this to other features
  - Explore *why* more thoroughly — why are these changes occurring?

Project 5: FCA for Plugin API Usage

- Recent work from ICPC 2016 started to look at plugin usage, search based strictly on text

- REU project goals

- Can we find collections of plugin extension points used as “features” that should be implemented together?

- Can we use this to help developers to find the correct features to implement, guide them to plugins and implementations they can use for guidance?

- Note: uses Formal Concept Analysis

Collaborators

- David Anderson (ECU)
- T. Baris Aktemur (Özyeğin University, Turkey)
- Marcelo d'Amorim (UFPE, Brazil)
- Jeroen van den Bos (CWI)
- Feng Chen (UIUC)
- Maurits Henneke (ippz)
- Paul Klint (CWI)
- Dimitris Kyritsis (UvA)
- Lindsey Lanier (ECU)
- Patrick Meredith (UIUC)
- Chris Mulder (UvA, Hyves)
- Grigore Rosu (UIUC)
- Ioana Rucareanu (UvA)
- Traian Serbanuta (UAIC, Romania)
- Tijs van der Storm (CWI)
- Apil Tamang (ECU)
- Frank Tip (Waterloo)
- Jurgen Vinju (CWI)

Related Work: Meta-Programming

- ASF+SDF (Klint et al)
- RScript (Klint et al)
- Spoofax (Visser et al)
- JastAdd (Hedin et al)
- TXL (Cordy)
- XText (itemis)
- CrocoPat (Beyer)
- Grok (Holt)
- JetBrains MPS (JetBrains)
- Kiama (Sloane)
- many others

Related Work: Analysis for Dynamic Languages

"Eval Begone!: Semi-Automated Removal of eval from JavaScript Programs", Fadi Meawad, Gregor Richards, Floréal Morandat, Jan Vitek. OOPSLA 2012.

"Tool-supported Refactoring for JavaScript", Asger Feldthaus, Todd D. Millstein, Anders Møller, Max Schäfer, Frank Tip. OOPSLA 2011.

"The Eval That Men Do - A Large-Scale Study of the Use of Eval in JavaScript Applications", Gregor Richards, Christian Hammer, Brian Burg, Jan Vitek. ECOOP 2011.

"An Analysis of the Dynamic Behavior of JavaScript Programs", Gregor Richards, Sylvain Lebresne, Brian Burg, Jan Vitek, PLDI 2010.

"Profile-Guided Static Typing for Dynamic Scripting Languages", Michael Furr, Jong-hoon (David) An, Jeffrey S. Foster. OOPSLA 2009.

"Type Analysis for JavaScript", Simon Holm Jensen, Anders Møller, Peter Thiemann. SAS 2009.

Related Work: Program Analysis for PHP

"The HipHop Compiler for PHP", Haiping Zhao, Iain Proctor, Minghui Yang, Xin Qi, Mark Williams, Qi Gao, Guilherme Ottoni, Andrew Paroski, Scott MacVicar,Jason Evans, Stephen Tu. OOPSLA 2012.

"Design and Implementation of an Ahead-of-Time Compiler for PHP", Paul Biggar. PhD Thesis, Trinity College Dublin, April 2010.

"Static Detection of Cross-Site Scripting Vulnerabilities", Gary Wassermann, Zhendong Su. ICSE 2008.

"Sound and Precise Analysis of Web Applications for Injection Vulnerabilities", Gary Wassermann, Zhendong Su. PLDI 2007.

Discussion

Thank you!
Any Questions?

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