Debunking Design Flaws in PHP Code using Static Call Graphs

Berlin PHP Usergroup
Falko Menge
07.11.2007

Agenda

- Motivation
- PHPCallGraph
- Results
- 3D Exploration with the CGA framework
- Conclusion

Motivation

- When working with large software systems:
  - Hard to get an overview of the system
  - High number of dependencies
  - Reading complete source code takes too much time
  - Even harder if its not your own code

- Automatic visualization of dependencies could help to handle the complexity

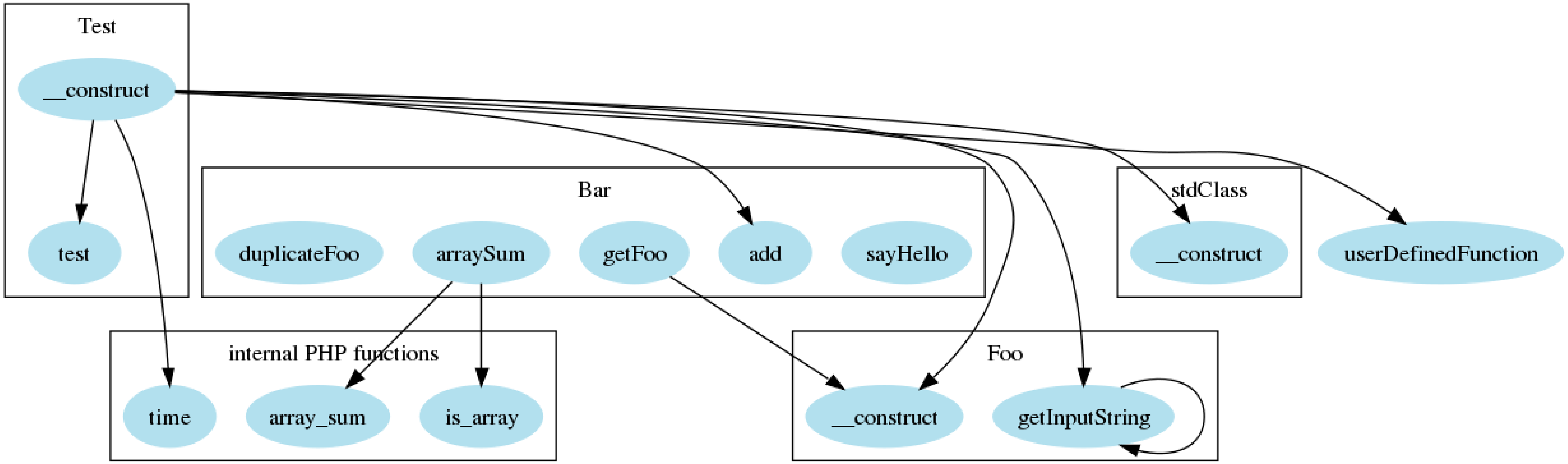
PHPCallGraph: First Prototype

- Static call graph generator for PHP
- 50 lines of PHP code
- Source code parsing with regular expressions
  - Lead to several bugs
- Graph rendering with DOT
  - Part of open source GraphViz framework for visualization of directed and undirected graphs

PHPCallGraph: Improvements

- Leveraging InstantSVC CodeAnalyzer
- Parsing of method bodies with PHP's Tokenizer
- DOT generation through PEAR package Image_GraphViz by Sebastian Bergmann
- ezcConsoleTools for command line frontend
- Output driver for 3D exploration with CGA

Results

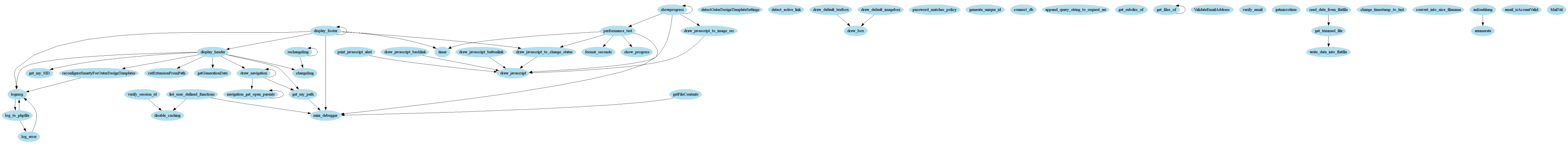


Test
__construct
test
Bar
duplicateFoo
arraySum
getFoo
add
sayHello
stdClass
__construct
userDefinedFunction
internal PHP functions
time
array_sum
is_array
Foo
__construct
getInputString

Results

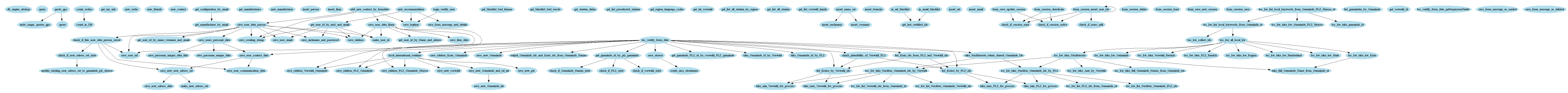
- Design flaws which can be detected
  - Cyclic dependencies
  - Dead code
  - Candidates for refactoring
    - Subclasses
    - Separation of concerns
    - Introduction of visibilities
      (especially when migrating from PHP4 to PHP5)

Identifying Candidates for Refactoring



- Real world example:
  - Function library of 55 functions
  - Nearly 2000 lines of code (90KB)
- Call graph shows lots of dependencies
  => Introduction of several classes

Identifying Candidates for Refactoring

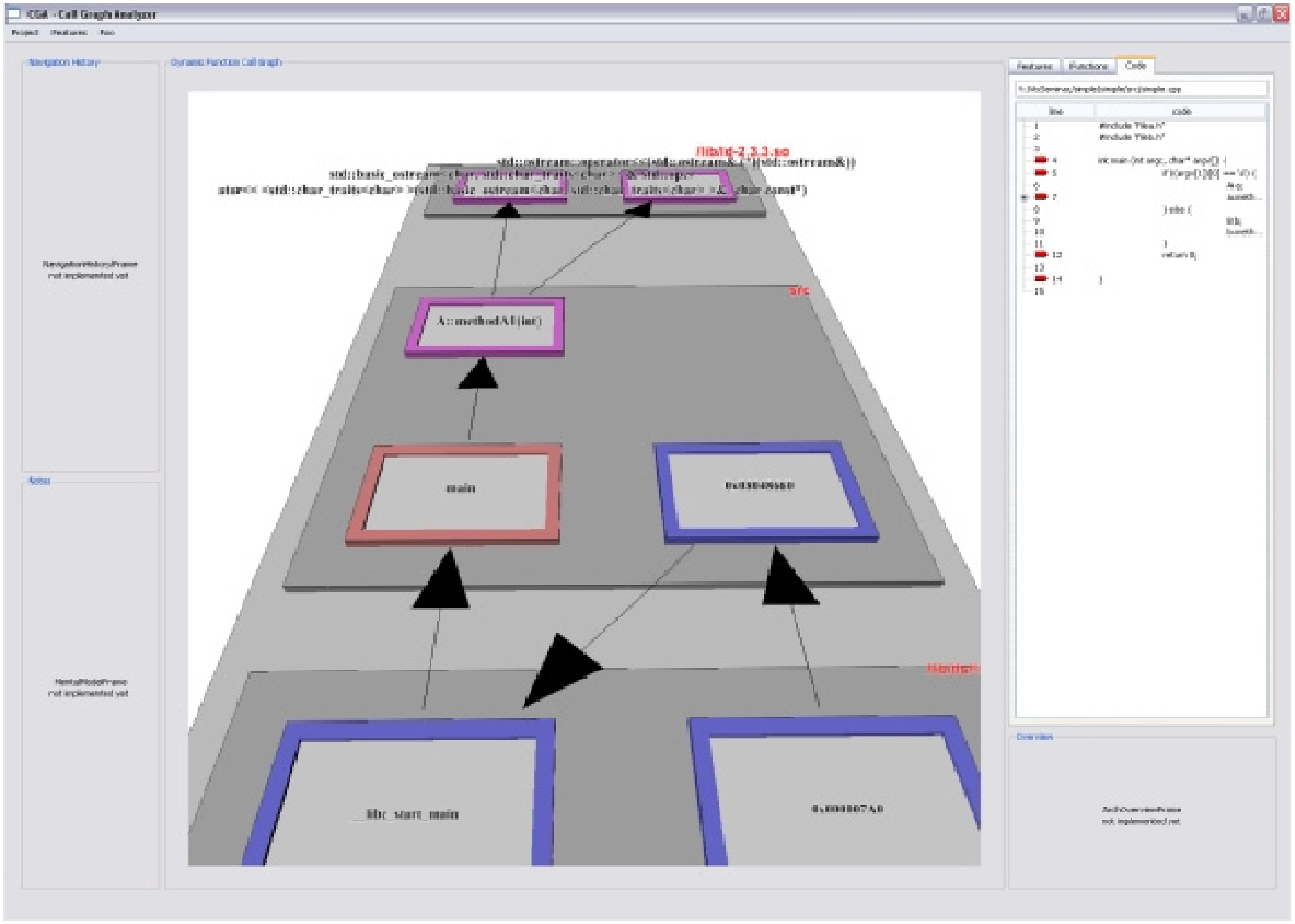


- Real world example:
  - One single class containing 130 methods
  - Over 5000 lines of code (190KB)
- Call graph shows clearly separated clusters => Separation into different classes

3D Exploration with CGA

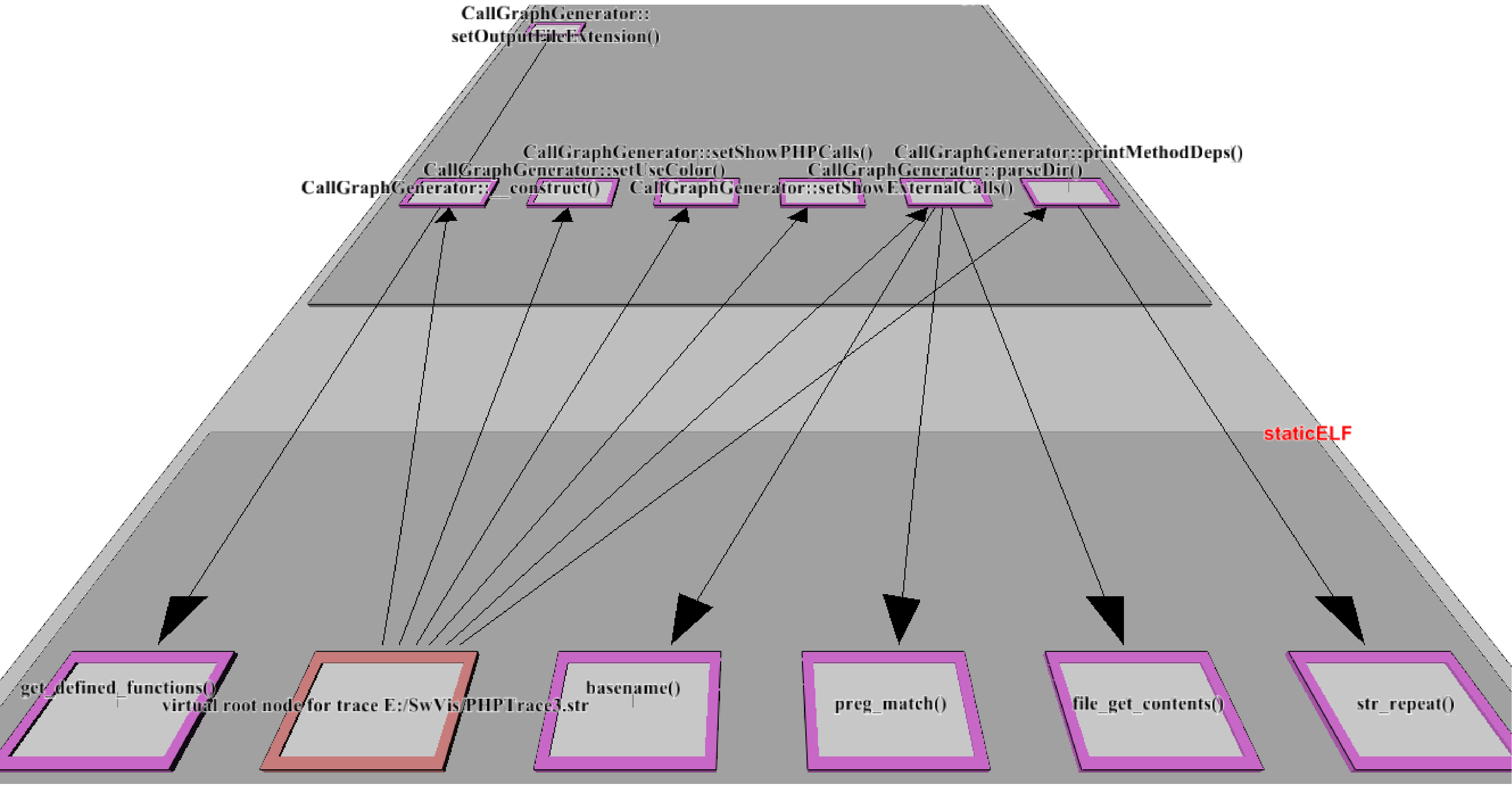
- Framework for analyzing complex software systems
- Focus on various aspects of system dynamics
- Provides elaborate visualization techniques
- Analysis of function level dynamics and long-term system evolution
- Developed by Computer Graphics System group of the Hasso Plattner Institute

3D Exploration with CGA



```
1   #include "A.h"
2   #include "B.h"
3
4   int main (int argc, char** argv[]) {
5           if (argv[1][0] == '0') {
6                   A a;
7                   a.meth...
8           } else {
9                   B b;
10                  b.meth...
11          }
12          return 0;
13
14  }
15
```

3D Exploration with CGA


CallGraphGenerator::
setOutputFileExtension()
CallGraphGenerator::setShowPHPCalls()
CallGraphGenerator::printMethodDeps()
CallGraphGenerator::setUseColor()
CallGraphGenerator::parseDir()
CallGraphGenerator::__construct()
CallGraphGenerator::setShowExternalCalls()
staticELF
get_defined_functions()
virtual root node for trace E:/SwVis/PHPTrace3.str
basename()
preg_match()
file_get_contents()
str_repeat()

Conclusion

- Static call graphs can be leveraged to gain a better understanding of large systems
- Various design flaws can be detected
- Reflection can be used for static analysis

http://phpcallgraph.sf.net

http://instantsvc.sf.net

http://cgs.hpi.uni-potsdam.de/trac/cga/