CSE 503

Software Engineering

Course introduction

Key questions:
What does your program do?
How do you know?

Today

- Logistics
- Brief introduction
- Course overview
- Why program analysis?

Logistics

- Course material, schedule, etc. on course website: https://courses.cs.washington.edu/courses/cse503/ All slides are posted before class.
- Assignment submission and discussions via Canvas (linked from webpage)

The CSE 503 team

Instructor

- Michael Ernst
- Office hours: After class and by appointment
- mernst@cs.washington.edu

Teaching assistant

- James Yoo
- Office hours: TBD
- jmsy@cs.washington.edu

Your background

Introduction and a very brief survey

- What is your research area (or area of interest)?
- How long have you been in the program?
- What is your SE background (programming languages, etc.)?

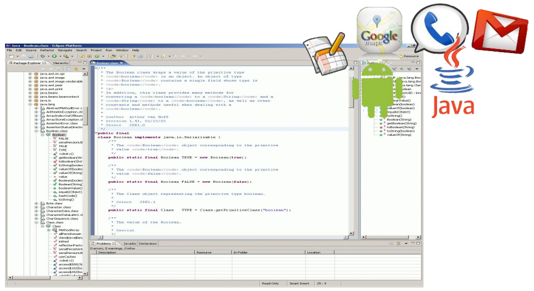
Today

- Logistics
- Brief introduction
- Course overview
- Why program analysis?

What is Software Engineering?

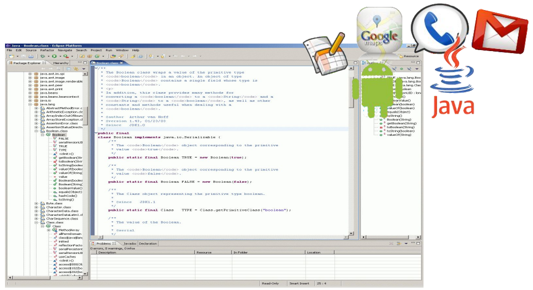
What is Software Engineering?

- Developing in an IDE
  and software ecosystem?



What is Software Engineering?

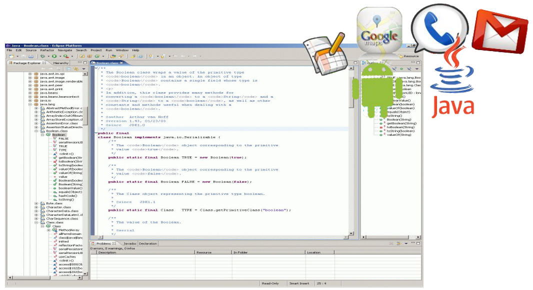
- Developing in an IDE
  and software ecosystem?



- Testing and debugging?

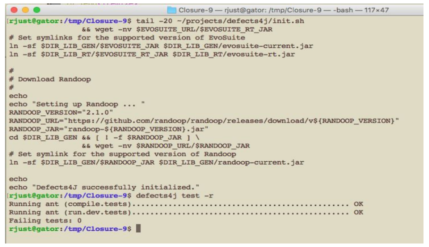
What is Software Engineering?

- Developing in an IDE
  and software ecosystem?



- Testing and debugging?

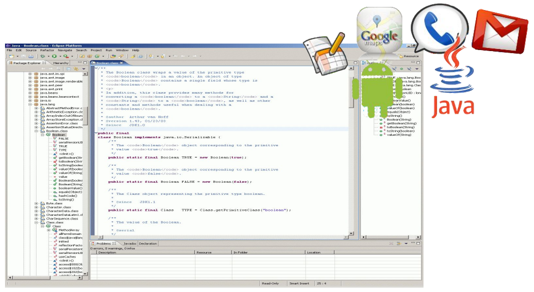
- Deploying and running
  a software system?

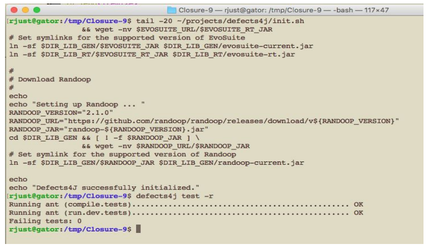


```
Closure-9 — rjust@gator: /tmp/Closure-9 — -bash — 117×47
rjust@gator:/tmp/Closure-9$ tail -20 ~/projects/defects4j/init.sh
                && wget -nv $EVOSUITE_URL/$EVOSUITE_RT_JAR
# Set symlinks for the supported version of EvoSuite
ln -sf $DIR_LIB_GEN/$EVOSUITE_JAR $DIR_LIB_GEN/evosuite-current.jar
ln -sf $DIR_LIB_RT/$EVOSUITE_RT_JAR $DIR_LIB_RT/evosuite-rt.jar

#
# Download Randoop
#
echo
echo "Setting up Randoop ... "
RANDOOP_VERSION="2.1.0"
RANDOOP_URL="https://github.com/randoop/randoop/releases/download/v${RANDOOP_VERSION}"
RANDOOP_JAR="randoop-${RANDOOP_VERSION}.jar"
cd $DIR_LIB_GEN && [ ! -f $RANDOOP_JAR ] \
                && wget -nv $RANDOOP_URL/$RANDOOP_JAR
# Set symlink for the supported version of Randoop
ln -sf $DIR_LIB_GEN/$RANDOOP_JAR $DIR_LIB_GEN/randoop-current.jar

echo
echo "Defects4J successfully initialized."
rjust@gator:/tmp/Closure-9$ defects4j test -r
Running ant (compile.tests).............................................. OK
Running ant (run.dev.tests).............................................. OK
Failing tests: 0
rjust@gator:/tmp/Closure-9$ 
```

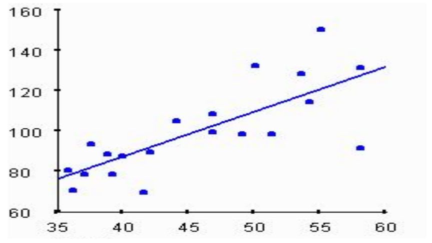
What is Software Engineering?

- Developing in an IDE and software ecosystem?
- Testing and debugging?
- Deploying and running a software system?
- Empirical evaluations?



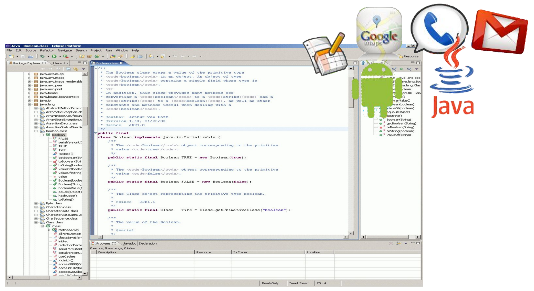

Closure-9 — rjust@gator: /tmp/Closure-9 — -bash — 117×47

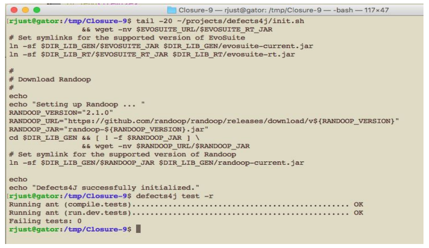
```
rjust@gator:/tmp/Closure-9$ tail -20 ~/projects/defects4j/init.sh
                && wget -nv $EVOSUITE_URL/$EVOSUITE_RT_JAR
# Set symlinks for the supported version of EvoSuite
ln -sf $DIR_LIB_GEN/$EVOSUITE_JAR $DIR_LIB_GEN/evosuite-current.jar
ln -sf $DIR_LIB_RT/$EVOSUITE_RT_JAR $DIR_LIB_RT/evosuite-rt.jar

#
# Download Randoop
#
echo
echo "Setting up Randoop ... "
RANDOOP_VERSION="2.1.0"
RANDOOP_URL="https://github.com/randoop/randoop/releases/download/v${RANDOOP_VERSION}"
RANDOOP_JAR="randoop-${RANDOOP_VERSION}.jar"
cd $DIR_LIB_GEN && [ ! -f $RANDOOP_JAR ] \
                && wget -nv $RANDOOP_URL/$RANDOOP_JAR
# Set symlink for the supported version of Randoop
ln -sf $DIR_LIB_GEN/$RANDOOP_JAR $DIR_LIB_GEN/randoop-current.jar

echo
echo "Defects4J successfully initialized."
rjust@gator:/tmp/Closure-9$ defects4j test -r
Running ant (compile.tests)................................................ OK
Running ant (run.dev.tests)................................................ OK
Failing tests: 0
rjust@gator:/tmp/Closure-9$ 
```



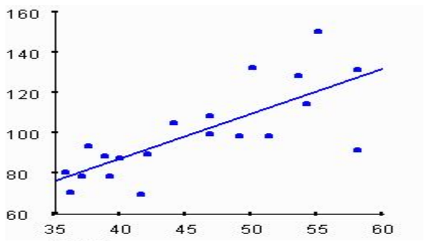
What is Software Engineering?

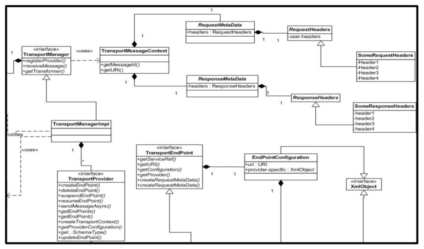
- Developing in an IDE and software ecosystem
- Testing and debugging
- Deploying and running a software system
- Empirical evaluations
- Modeling and designing





```
rjust@gator:/tmp/Closure-9$ tail -20 ~/projects/defects4j/init.sh
                && wget -nv $EVOSUITE_URL/$EVOSUITE_RT_JAR
# Set symlinks for the supported version of EvoSuite
ln -sf $DIR_LIB_GEN/$EVOSUITE_JAR $DIR_LIB_GEN/evosuite-current.jar
ln -sf $DIR_LIB_RT/$EVOSUITE_RT_JAR $DIR_LIB_RT/evosuite-rt.jar

#
# Download Randoop
#
echo
echo "Setting up Randoop ... "
RANDOOP_VERSION="2.1.0"
RANDOOP_URL="https://github.com/randoop/randoop/releases/download/v${RANDOOP_VERSION}"
RANDOOP_JAR="randoop-${RANDOOP_VERSION}.jar"
cd $DIR_LIB_GEN && [ ! -f $RANDOOP_JAR ] \
                && wget -nv $RANDOOP_URL/$RANDOOP_JAR
# Set symlink for the supported version of Randoop
ln -sf $DIR_LIB_GEN/$RANDOOP_JAR $DIR_LIB_GEN/randoop-current.jar

echo
echo "Defects4J successfully initialized."
rjust@gator:/tmp/Closure-9$ defects4j test -r
Running ant (compile.tests)............................................... OK
Running ant (run.dev.tests)............................................... OK
Failing tests: 0
rjust@gator:/tmp/Closure-9$ 
```





What is Software Engineering?

More than just writing code

The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
    - Requirements engineering
    - Specification writing and documentation
    - Software architecture and design
    - Programming
    - Software testing and debugging
    - Refactoring

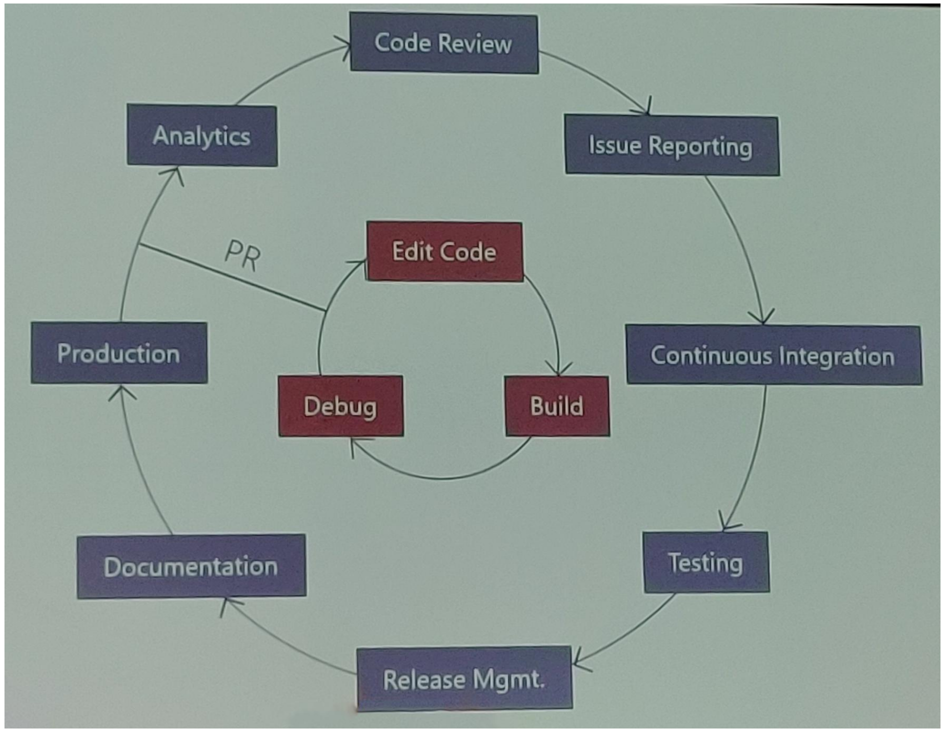
What is Software Engineering?

More than just writing code

The complete process of specifying, designing, developing, analyzing, deploying, and maintaining a software system.

- Common Software Engineering tasks include:
  - Requirements engineering
  - Specification writing and documentation
  - Software architecture and design
  - Programming — Just one out of many important tasks!
  - Software testing and debugging
  - Refactoring

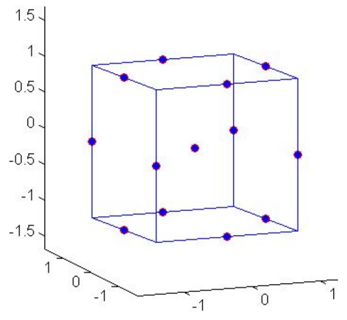
The Role of Software Engineering in Practice



(Development workflow at Microsoft, Big Code summit 2019)

The Role of Software Engineering in Research

Experimental infrastructure is software, too!



Infrastructure

Design space exploration

1	0.34	0.81
2	0.52	0.32
3	0.21	0.53
4	0.81	0.22
...	...	...

Example (automated debugging)

- 150 configurations, 1000+ benchmarks
- 1-85 hours per execution
- 200,000+ CPU hours (~23 CPU years)

Course overview: the big picture

- **Week 1:** Introduction & static vs. dynamic analysis
- **Week 2:** Symbolic reasoning
- **Week 3:** Symbolic reasoning
- **Week 4:** Testing
- **Week 5:** Delta Debugging
- **Week 6:** Invariants
- **Week 7:** Program Repair
- **Week 8:** Empirical Software Engineering
- **Week 9:** ML for Software Engineering
- **Week 10:** Wrap up

Course overview: the big picture

- **Week 1:** Introduction & static vs. dynamic analysis **HW 1**
- **Week 2:** Symbolic reasoning
- **Week 3:** Symbolic reasoning **HW 2**

formal, static

- **Week 4:** Testing
- **Week 5:** Delta Debugging **In-class exercise**
- **Week 6:** Invariants

depends on executions

- **Week 7:** Program Repair
- **Week 8:** Empirical Software Engineering
- **Week 9:** ML for Software Engineering

depends on heuristics

- **Week 10:** Wrap up **Project presentation**

Questions?

Course overview: this week

- **Week 1:** Introduction & static vs. dynamic analysis **HW 1**
- **Two high-level papers**
  - Static and dynamic analysis: synergy and duality
  - Lessons from building static analysis tools at Google
- **HW 1**
  - Brainstorming about software development difficulties
  - **Please** start right away!

Course overview: the project

Logistics

- 2-4 team members
- Synergies with **your** work are welcome!

Timeline

- **Week 3/4**: Project proposal and revision
- **Week 6**: Related work and methodology
- **Week 8**: Coding completed and initial results
- **Week 10**: Presentation and final report

Course overview: the project

Logistics

- 2-4 team members
- Synergies with **your** work are welcome!

Timeline

- **Week 3/4**: Project proposal and revision
- **Week 6**: Related work and methodology
- **Week 8**: Coding completed and initial results
- **Week 10**: Presentation and final report

Types of projects (non-exhaustive)

- proposing and evaluating a new technique
- developing and assessing new algorithms to replace currently-used ones
- translating a methodology to a new problem domain
- applying known techniques to new problem domains
- evaluation of existing techniques or tools (case studies or controlled experiment)
- implementation of a proposed but never implemented technique

Questions?

Course overview: the big picture

● **Week 1:** Introduction & static vs. dynamic analysis	**HW 1**
● **Week 2:** Symbolic reasoning	
● **Week 3:** Symbolic reasoning	**HW 2**
● **Week 4:** Testing	
● **Week 5:** Delta Debugging	**In-class exercise**
● **Week 6:** Invariants	
● **Week 7:** Program Repair	
● **Week 8:** Empirical Software Engineering	
● **Week 9:** ML for Software Engineering	
● **Week 10:** Wrap up	**Project presentation**

And there is more...

Special topics:

- **504: AI meets Software engineering**
  (ML and statistical methods for SE/program analysis)
- **599: Research methods**
  (Research design and statistics in R)

Course overview: the big picture

● **Week 1:** Introduction & static vs. dynamic analysis	**HW 1**
● **Week 2:** Symbolic reasoning	
● **Week 3:** Symbolic reasoning	**HW 2**
● **Week 4:** Testing	
● **Week 5:** Delta Debugging	**In-class exercise**
● **Week 6:** Invariants	
● **Week 7:** Program Repair	
● **Week 8:** Empirical Software Engineering	
● **Week 9:** ML for Software Engineering	
● **Week 10:** Wrap up	**Project presentation**

Course overview: grading

- **50%** Class project
- **35%** HWs, in-class exercise, reading questions
- **15%** Participation

Questions?

Course overview: expectations

- Conducting a quarter-long research project
- Some programming experience
- Reading and actively discussing research papers
- Have fun!

Who can be successful?

- **You** can!
- Assumes an undergraduate CS education (= 1st year grad)
- You will *learn* to read papers, write papers, conduct research, etc. That is a goal of the class.
- Ask lots of questions, so we can help you

Today

- Logistics
- Brief introduction
- Your background
- Course overview
- Why program analysis?

Who cares about program analysis?

Who cares about program analysis?

- ~15 million lines of code

Let’s say 50 lines per page (0.05 mm)

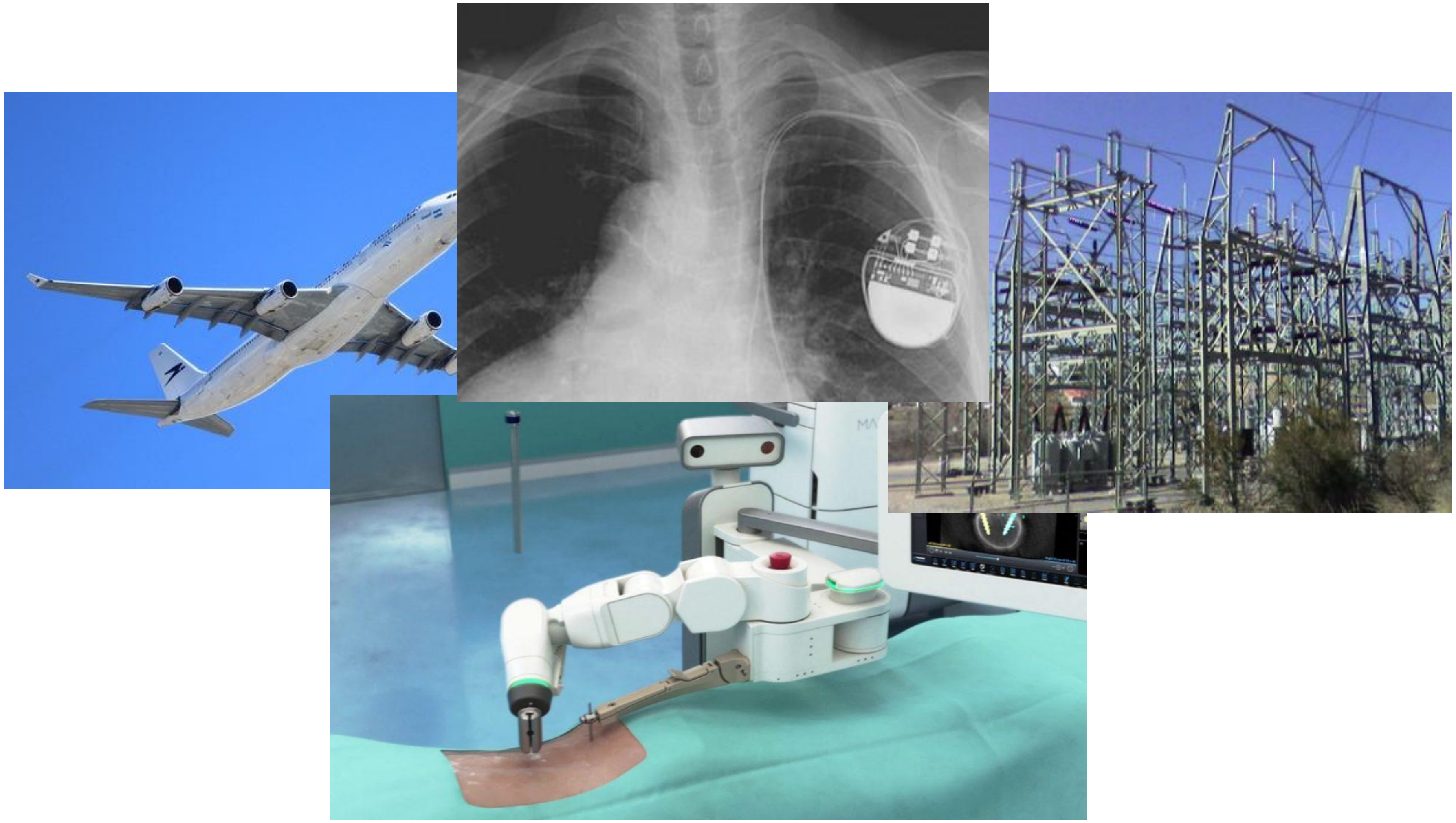
Who cares about program analysis?

- ~15 million lines of code

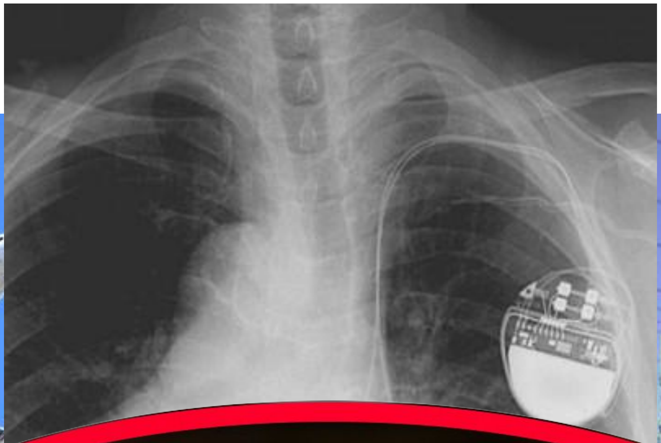
Let's say 50 lines per page

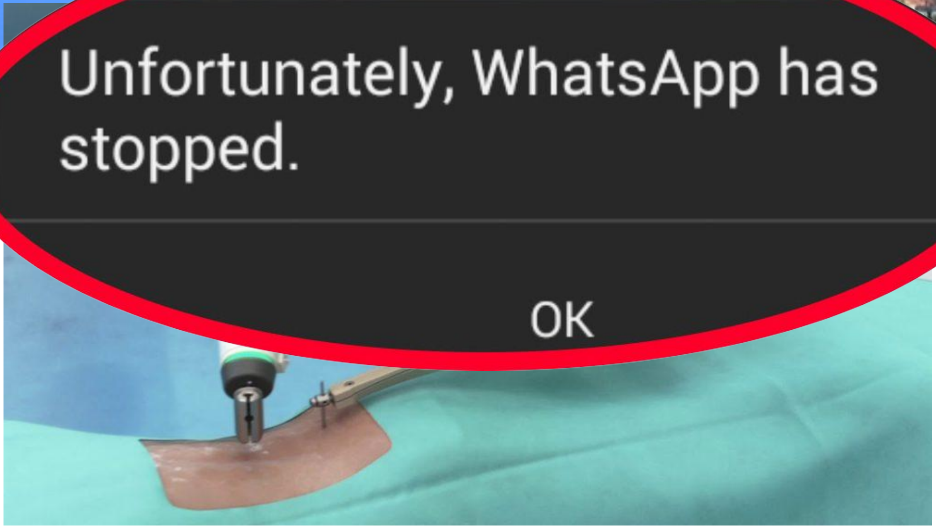
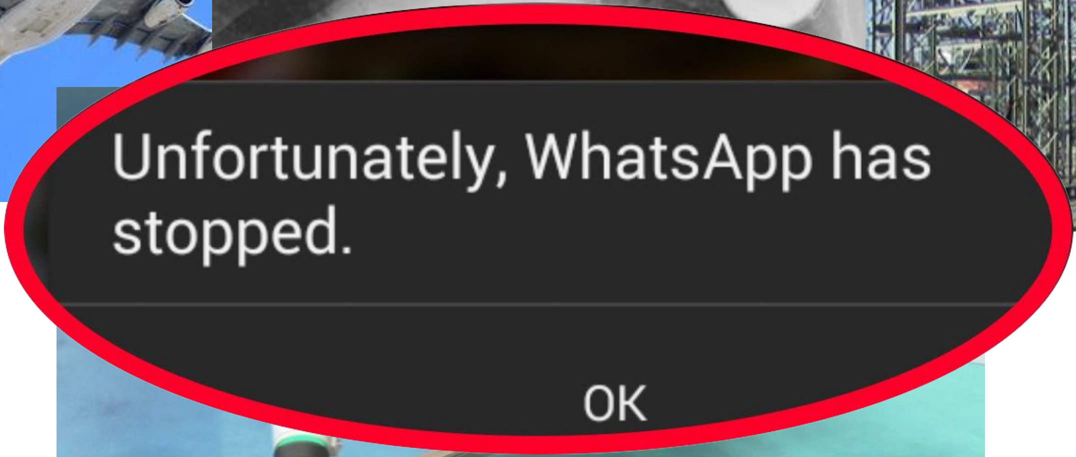
- 300000 pages
- 15 m (49 ft)

Who cares about program analysis?



Who cares about program analysis?

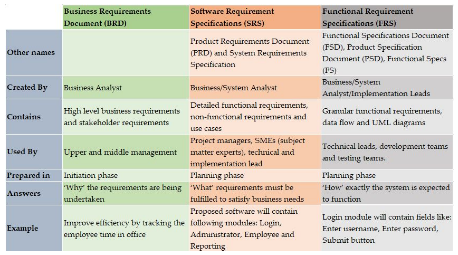




Unfortunately, WhatsApp has stopped.

Program analysis: examples

Does my program implement its specification?

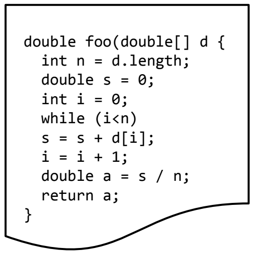
```
double foo(double[] d {
  int n = d.length;
  double s = 0;
  int i = 0;
  while (i<n)
  s = s + d[i];
  i = i + 1;
  double a = s / n;
  return a;
}
```



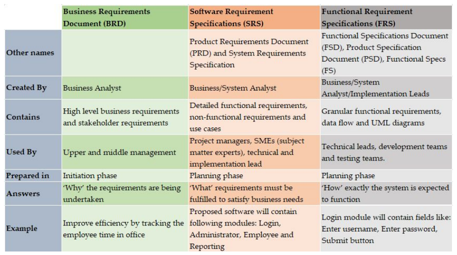
	Business Requirements Document (BRD)	Software Requirement Specifications (SRS)	Functional Requirement Specifications (FRS)
Other names		Product Requirements Document (PRD) and System Requirements Specification	Functional Specifications Document (FSD), Product Specification Document (PSD), Functional Specs (FS)
Created By	Business Analyst	Business/System Analyst	Business/System Analyst/Implementation Leads
Contains	High level business requirements and stakeholder requirements	Detailed functional requirements, non-functional requirements and use cases	Granular functional requirements, data flow and UML diagrams
Used By	Upper and middle management	Project managers, SMEs (subject matter experts), technical and implementation lead	Technical leads, development teams and testing teams.
Prepared in	Initiation phase	Planning phase	Planning phase
Answers	'Why' the requirements are being undertaken	'What' requirements must be fulfilled to satisfy business needs	'How' exactly the system is expected to function
Example	Improve efficiency by tracking the employee time in office	Proposed software will contain following modules: Login, Administrator, Employee and Reporting	Login module will contain fields like: Enter username, Enter password, Submit button

Program analysis: examples

Does my program implement its specification?



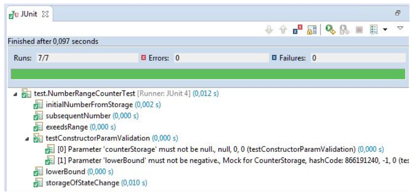
```
double foo(double[] d {
  int n = d.length;
  double s = 0;
  int i = 0;
  while (i<n)
  s = s + d[i];
  i = i + 1;
  double a = s / n;
  return a;
}
```



	Business Requirements Document (BRD)	Software Requirement Specifications (SRS)	Functional Requirement Specifications (FRS)
Other names		Product Requirements Document (PRD) and System Requirements Specification	Functional Specifications Document (FSD), Product Specification Document (PSD), Functional Specs (FS)
Created By	Business Analyst	Business/System Analyst	Business/System Analyst/Implementation Leads
Contains	High level business requirements and stakeholder requirements	Detailed functional requirements, non-functional requirements and use cases	Granular functional requirements, data flow and UML diagrams
Used By	Upper and middle management	Project managers, SMEs (subject matter experts), technical and implementation lead	Technical leads, development teams and testing teams.
Prepared in	Initiation phase	Planning phase	Planning phase
Answers	'Why' the requirements are being undertaken	'What' requirements must be fulfilled to satisfy business needs	'How' exactly the system is expected to function
Example	Improve efficiency by tracking the employee time in office	Proposed software will contain following modules: Login, Administrator, Employee and Reporting	Login module will contain fields like: Enter username, Enter password, Submit button

Example analyses

- Unit testing



```
Finished after 0,097 seconds
Runs: 7/7        Errors: 0        Failures: 0
test.NumberRangeCounterTest [Runner: JUnit 4] (0,012 s)
    initialNumberFromStorage (0,002 s)
    subsequentNumber (0,000 s)
    exeedsRange (0,000 s)
    testConstructorParamValidation (0,000 s)
        [0] Parameter 'counterStorage' must not be null., null, 0, 0 (testConstructorParamValidation) (0,000 s)
        [1] Parameter 'lowerBound' must not be negative., Mock for CounterStorage, hashCode: 866191240, -1, 0 (tes
    lowerBound (0,000 s)
    storageOfStateChange (0,010 s)
```

- Solver-aided reasoning

$$(\forall x\ \mathrm{fsa}(x)) \Rightarrow (\exists y\ \mathrm{pda}(y) \wedge \mathrm{equivalent}(x,y))$$

Program analysis: examples

What does this program (binary) do?

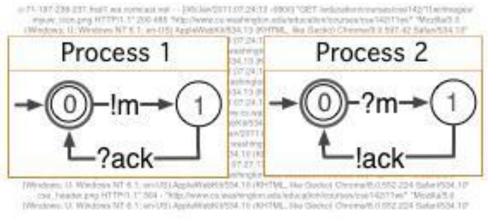
Program analysis: examples

What does this program (binary) do?

Example analyses

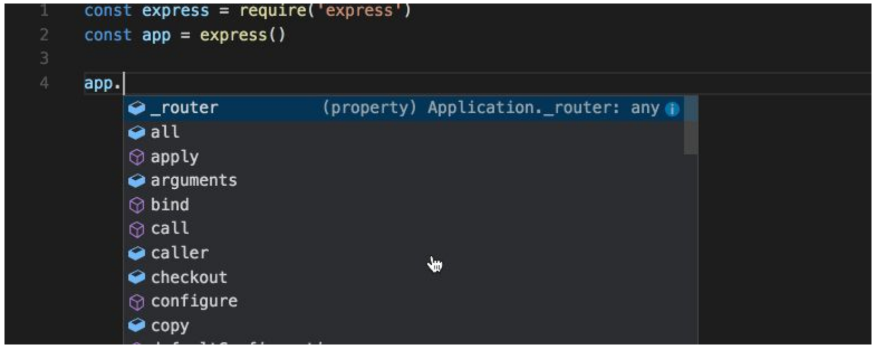
- Fuzzing

- Statistical inference of invariants and models



Program analysis: examples

Autocompletion: which methods to suggest?

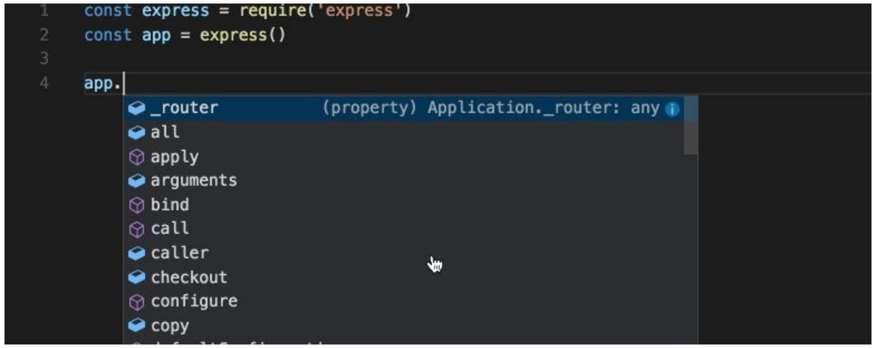


```
const express = require('express')
const app = express()

app.
```

```
_router          (property) Application._router: any
all
apply
arguments
bind
call
caller
checkout
configure
copy
```

Program analysis: examples

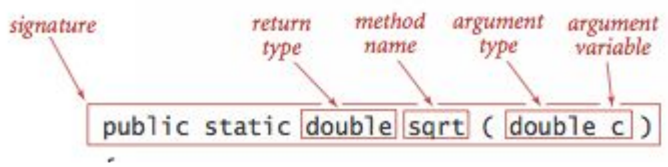
Autocompletion: which methods to suggest?

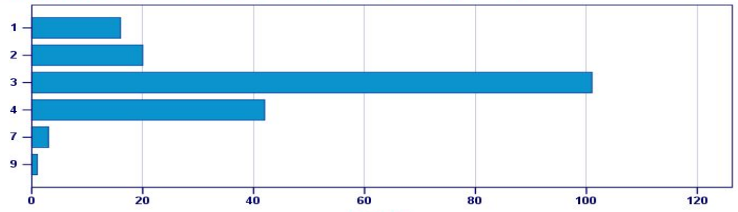


```
const express = require('express')
const app = express()

app.
```

Example analyses

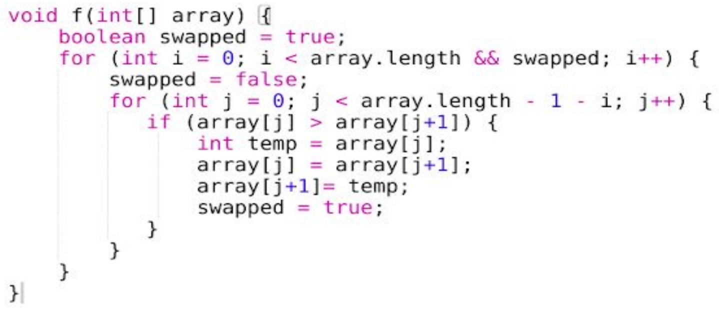
- Context-sensitive type checking
- Heuristics and frequency analysis





Program analysis: examples

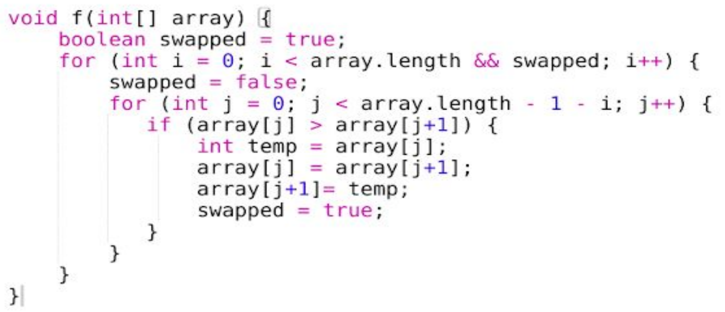
Semantics: how to name this method?



```
void f(int[] array) {
    boolean swapped = true;
    for (int i = 0; i < array.length && swapped; i++) {
        swapped = false;
        for (int j = 0; j < array.length - 1 - i; j++) {
           if (array[j] > array[j+1]) {
               int temp = array[j];
               array[j] = array[j+1];
               array[j+1]= temp;
               swapped = true;
           }
        }
    }
}
```

Program analysis: examples

Semantics: how to name this method?

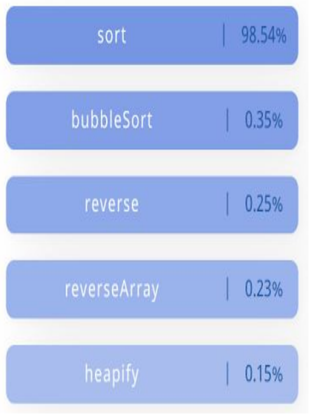


```
void f(int[] array) {
    boolean swapped = true;
    for (int i = 0; i < array.length && swapped; i++) {
        swapped = false;
        for (int j = 0; j < array.length - 1 - i; j++) {
            if (array[j] > array[j+1]) {
                int temp = array[j];
                array[j] = array[j+1];
                array[j+1]= temp;
                swapped = true;
            }
        }
    }
}
```

Example analyses

- Statistical language models (bag of words, n-grams, etc.)

- Heuristics and frequency analysis



sort	98.54%
bubbleSort	0.35%
reverse	0.25%
reverseArray	0.23%
heapify	0.15%

Next time: static vs. dynamic analysis

A **static analysis** analyzes program source code without running the program

- What are examples?

A **dynamic analysis** observes program executions

- What are examples?

Under what circumstances is each one preferable?