

# CSE 503

Software Engineering

Winter 2021

**Course introduction**

January 06, 2021

# Today

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

# Logistics

- Wed/Fri, 10:00am – 11:20am.
- Lectures, discussions, and presentations via Zoom.
- Course material, schedule, etc. on course website:  
<https://homes.cs.washington.edu/~rjust/courses/2021Winter/CSE503>
- Submission of assignments via Canvas:  
<https://canvas.uw.edu>
- Communication via Slack:  
<https://cse503workspace.slack.com>

# The CSE 503 team

## Instructor

- René Just
- Office hours: After class and by appointment
- [rjust@cs.washington.edu](mailto:rjust@cs.washington.edu)

## Teaching assistant

- Brendan Wallace
- Office hours: TBD
- [bwbw@uw.edu](mailto:bwbw@uw.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

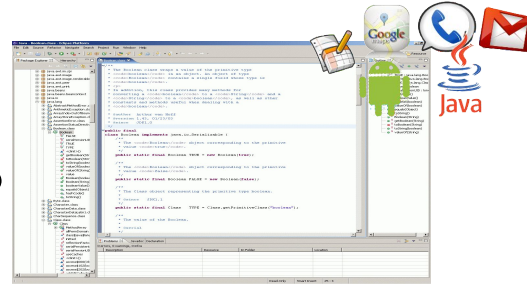
- Logistics
- Brief introduction
- Course overview
- Why program analysis?

# What is Software Engineering?



# What is Software Engineering?

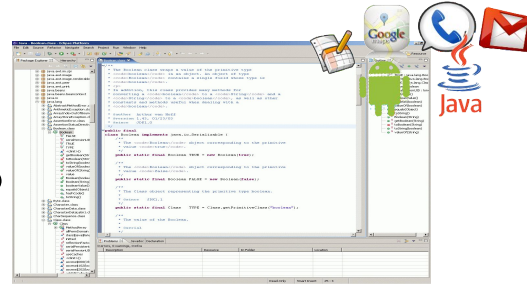
- Developing in an IDE and software ecosystem?





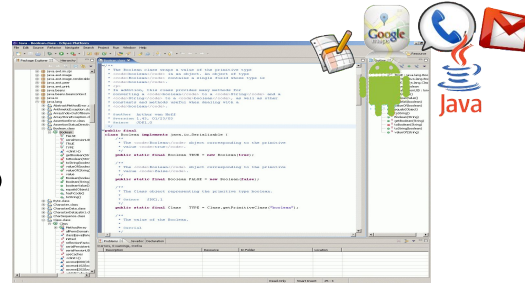
# What is Software Engineering?

- 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?

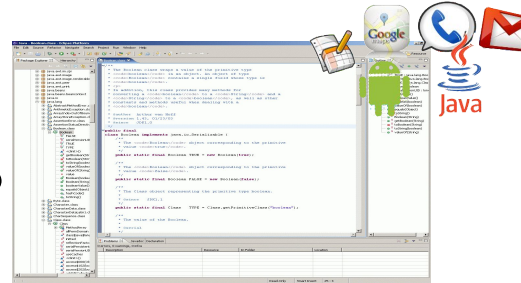


```
Close-0 --- Just@gator:/tmp/Close-0 --- bash -- 117.47
~/defects4j
$ java -jar evosuite-rt.jar
# set symlink for the supported version of Boodilla
ln -sf $DIR_LIB_GEN/evosuite.jar $DIR_LIB_GEN/evosuite-current.jar
ln -sf $DIR_LIB_RT/evosuite-rt.jar $DIR_LIB_RT/evosuite-rt-current.jar
#
# Download Randoop
#
echo "Setting up Randoop ..."
RANDOOP_VERSION="2.1.0"
RANDOOP_URL="https://github.com/randoop/randoop/releases/download/v${RANDOOP_VERSION}"
RANDOOP_JAR="randoop-${RANDOOP_VERSION}.jar"
od $DIR_LIB_GEN && [ ! -f $RANDOOP_JAR ]
&& wget -Ov $RANDOOP_URL/$RANDOOP_JAR
# set symlink for the supported version of Randoop
ln -sf $DIR_LIB_GEN/$RANDOOP_JAR $DIR_LIB_GEN/randoop-current.jar

echo "Defects4J successfully initialized."
~/defects4j
$ java -jar evosuite-rt.jar
Running ant (compile.tests)..... OK
Running ant (run.dev.tests)..... OK
Falling testat: 0
~/defects4j
```

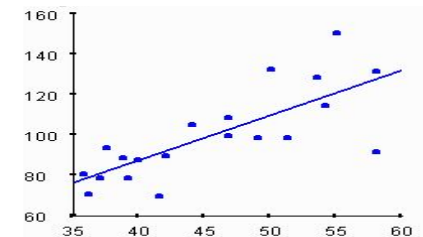
# What is Software Engineering?

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



```
Close-0 --- Just@gator:/tmp/Close-0 --- bash -- 117.47
~/Projects/Defects4J/init.sh
44 wget -nv $EVOBSITE_URL/$EVOBSITE_RT_JAR
# Set symlink for the supported version of Boodata
ln -sf $DIR_LIB_GEN/$EVOBSITE_RT_JAR $DIR_LIB_GEN/evoosite-current.jar
ln -sf $DIR_LIB_RT/$EVOBSITE_RT_JAR $DIR_LIB_RT/evoosite-rt.jar
#
# Download Randoop
#
echo "Setting up Randoop ..."
RANDOOP_VERSION="2.1.0"
RANDOOP_URL="https://github.com/randoop/randoop/releases/download/v${RANDOOP_VERSION}"
RANDOOP_JAR="randoop-${RANDOOP_VERSION}.jar"
od $DIR_LIB_GEN 44 { | -f $RANDOOP_JAR }
44 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 "Defects4J successfully initialized."
~/Projects/Defects4J/defects4j/test -s
Running ant (compile.tests)..... OK
Running ant (run.dev.tests)..... OK
Falling testat: 0
~/Projects/Defects4J
```





# 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

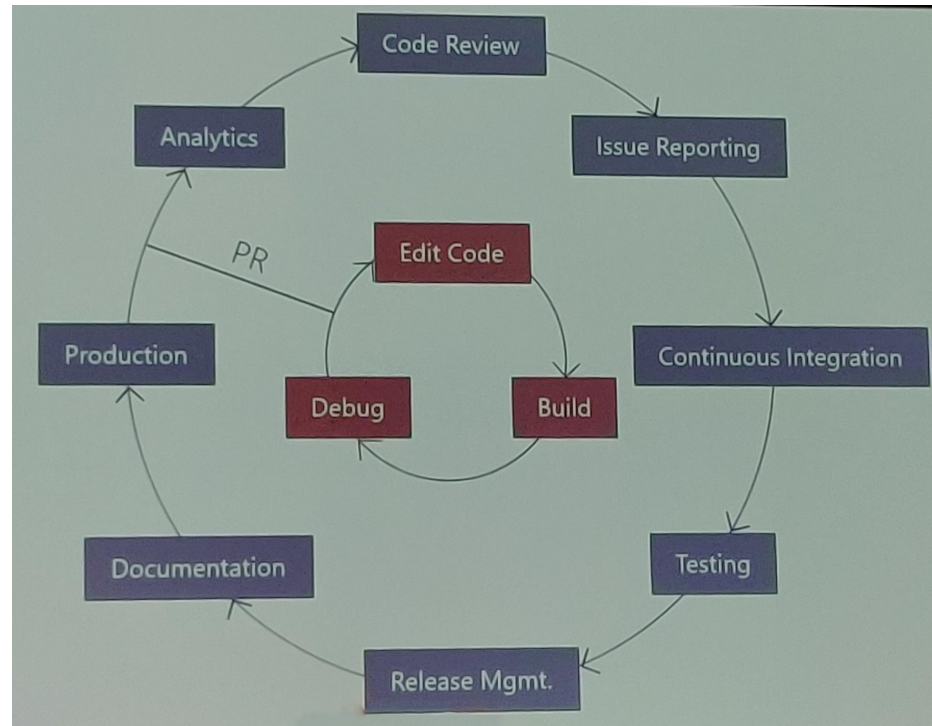
# What is Software Engineering?

## 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

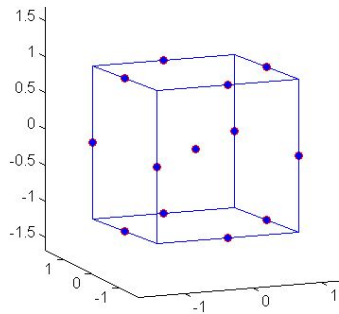
# The Role of Software Engineering in Practice



(Development workflow at Microsoft, Big Code summit 2019)

# The Role of Software Engineering in Research

## Experimental infrastructure is software, too!



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:** Abstract Interpretation
- **Week 3:** Abstract Interpretation
- **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:** Abstract Interpretation
- **Week 3:** Abstract Interpretation **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**

**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

# Course overview: the project

## Logistics

- 2-4 team members
- Synergies with **your** work are welcome! (Project ideas provided after HW 1)

## 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! (Project ideas provided after HW 1)

## 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 fundamental 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:** Abstract Interpretation
- **Week 3:** Abstract Interpretation **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**  
(Statistics and R done wrong)



# Course overview: the big picture

- **Week 1:** Introduction & static vs. dynamic analysis **HW 1**
- **Week 2:** Abstract Interpretation
- **Week 3:** Abstract Interpretation **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!

# 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?



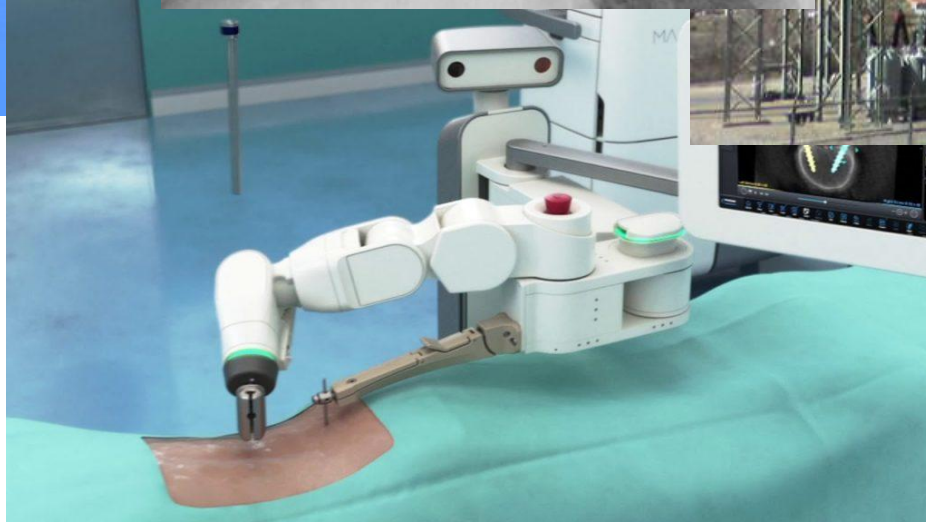
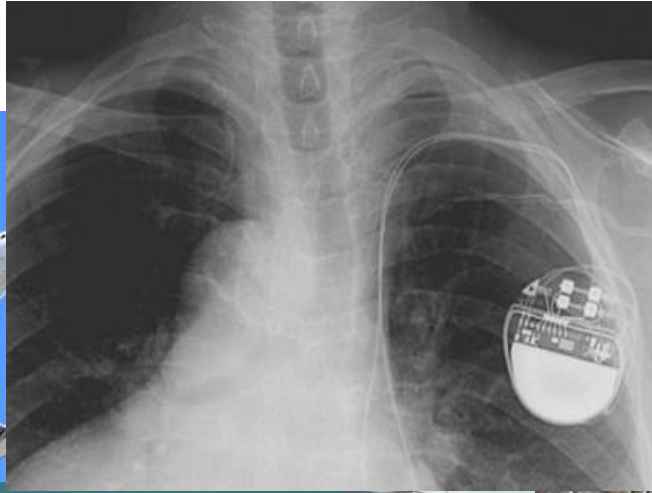
- ~15 million lines of code

Let's say 50 lines per page (0.05 mm)

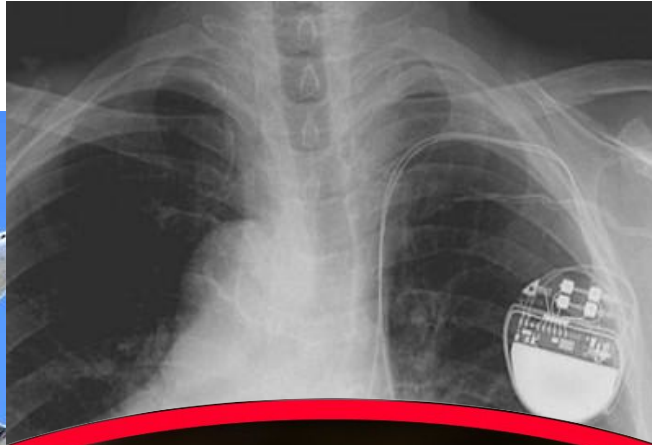
- 300000 pages
- 15 m (49 ft)



# Who cares about program analysis?

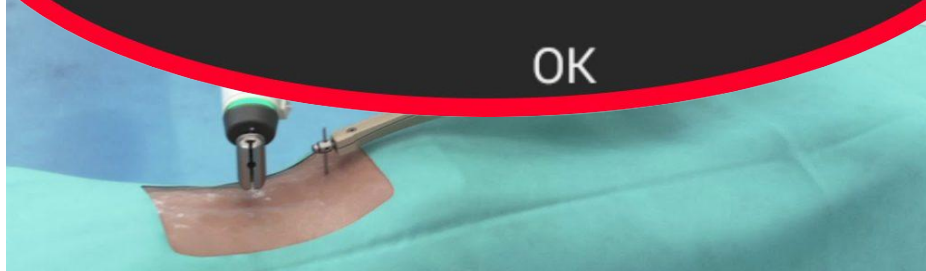


# Who cares about program analysis?



Unfortunately, WhatsApp has stopped.

OK





# 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?

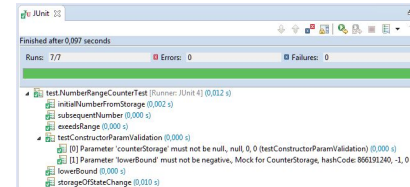
```
double foo(double[] d {
  int n = d.length;
  double s = 0;
  int i = 0;
  while (i < n)
    s = s + d[i];
  i = i + 1;
  double a = s / n;
  return a;
}
```



	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



- Solver-aided reasoning



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

# Program analysis: examples



**What does this program (binary) do?**



# Program analysis: examples

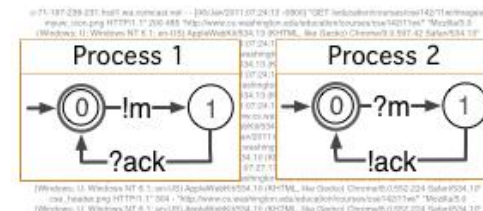
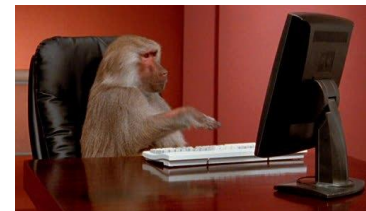


What does this program (binary) do?



## Example analyses

- Fuzzing
- Statistical inference of invariants and models



# Program analysis: examples



## Autocompletion: which methods to suggest?

```
1 const express = require('express')
2 const app = express()
3
4 app.|
```

- ⌵ \_router (property) Application.\_router: any ⓘ
- ⌵ all
- ⌵ apply
- ⌵ arguments
- ⌵ bind
- ⌵ call
- ⌵ caller
- ⌵ checkout
- ⌵ configure
- ⌵ copy

# Program analysis: examples

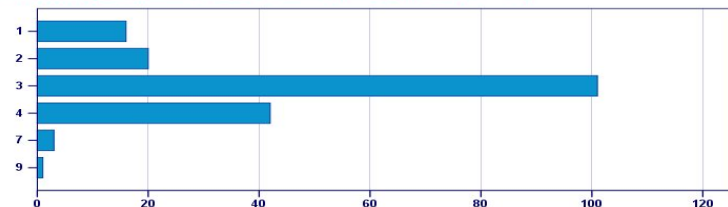
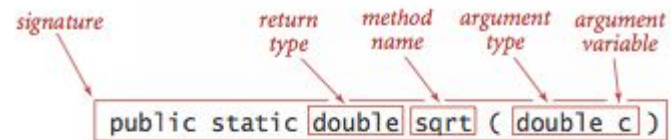


## Autocompletion: which methods to suggest?

```
1 const express = require('express')
2 const app = express()
3
4 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%