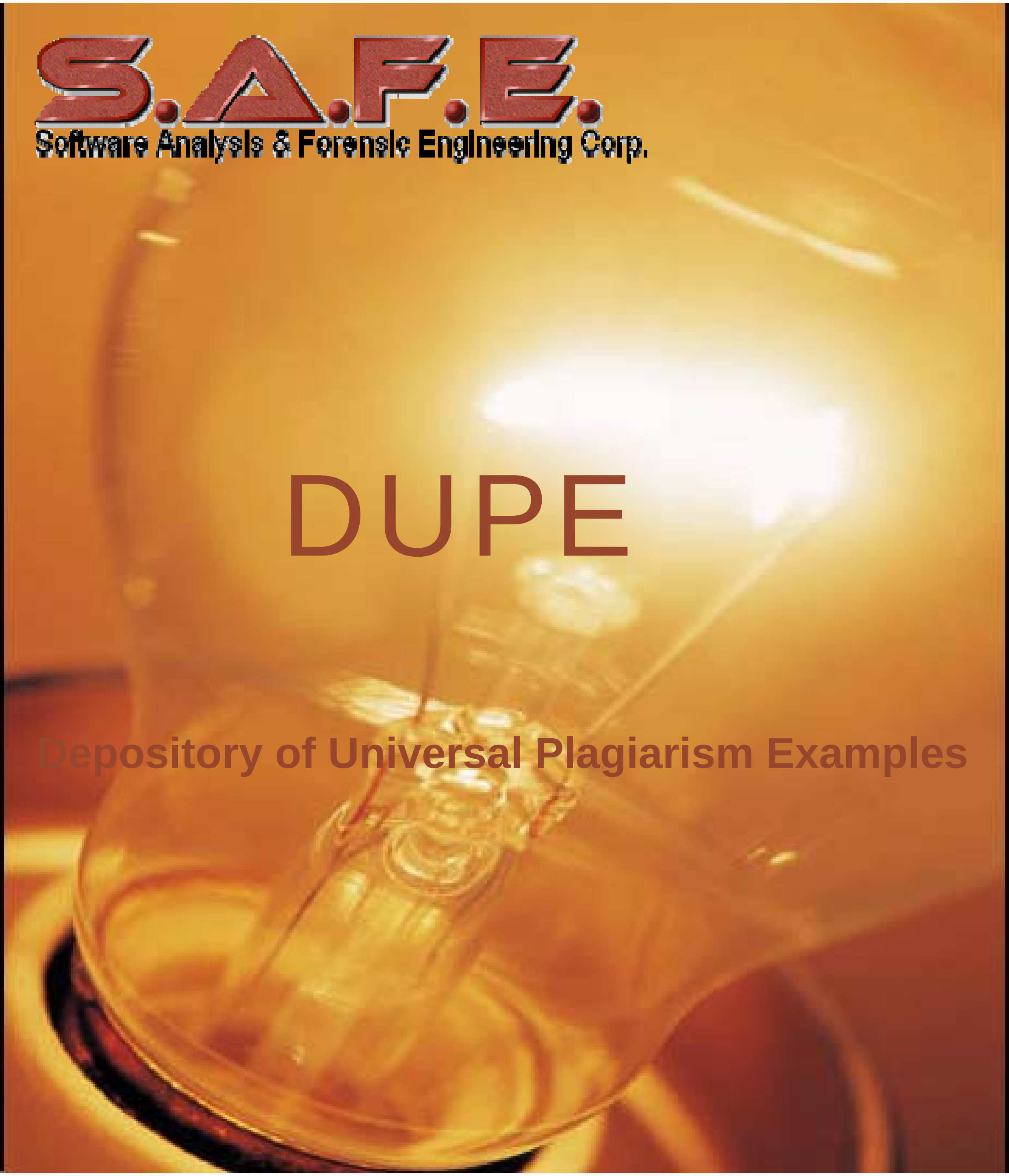
S.A.F.E.

Software Analysis & Forensic Engineering Corp.



# DUPE

Depository of Universal Plagiarism Examples

## Agenda

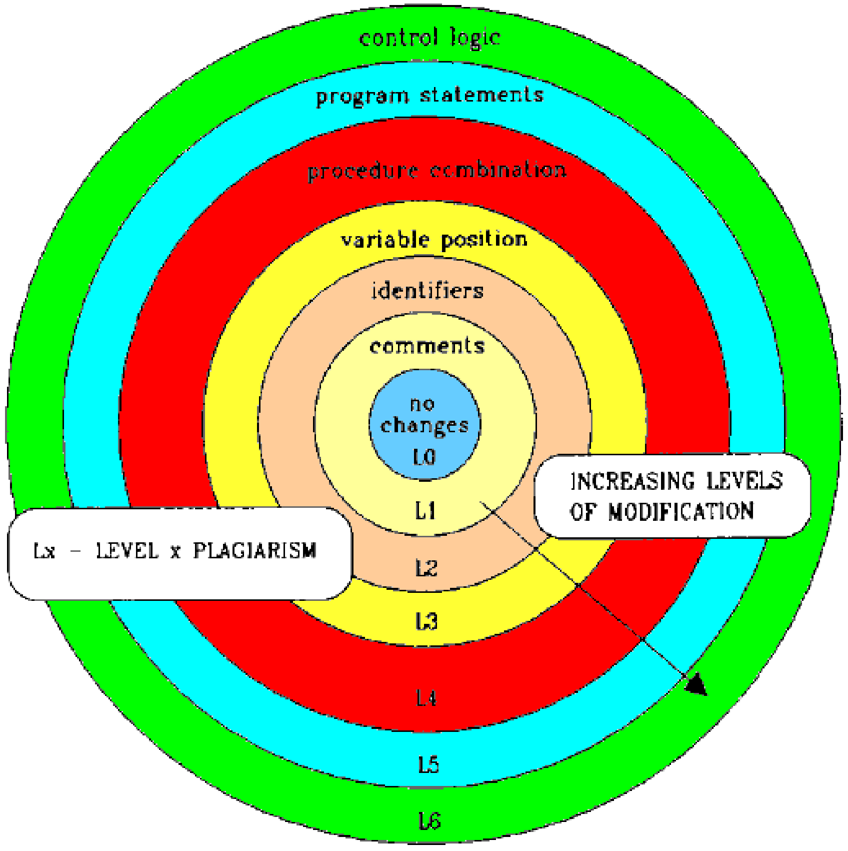
- Software Plagiarism
  - Definition
  - Measurement
- S.A.F.E. Tests
- Depository of Universal Plagiarism Examples
  - Choose open source projects
  - Definition of software plagiarism
  - Logistics
  - Legal issues
- Discussion/Partners

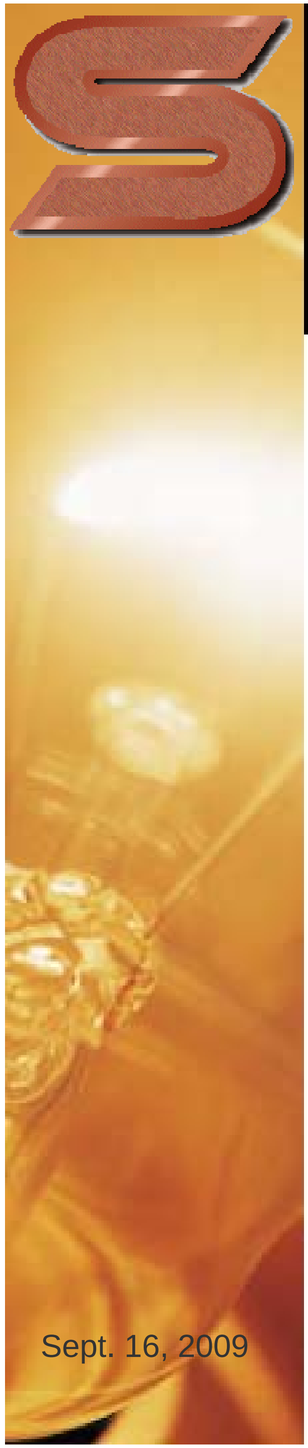
## Software Plagiarism

- Faidhi and Robinson
  - “An empirical approach for detecting program similarity and plagiarism within a university programming environment”, *Computer Education* Vol. 11. pp. 11-19, 1987.
- Six levels of program modification

# Plagiarism Measurement

- Faidhi and Robinson





## Plagiarism Measurement

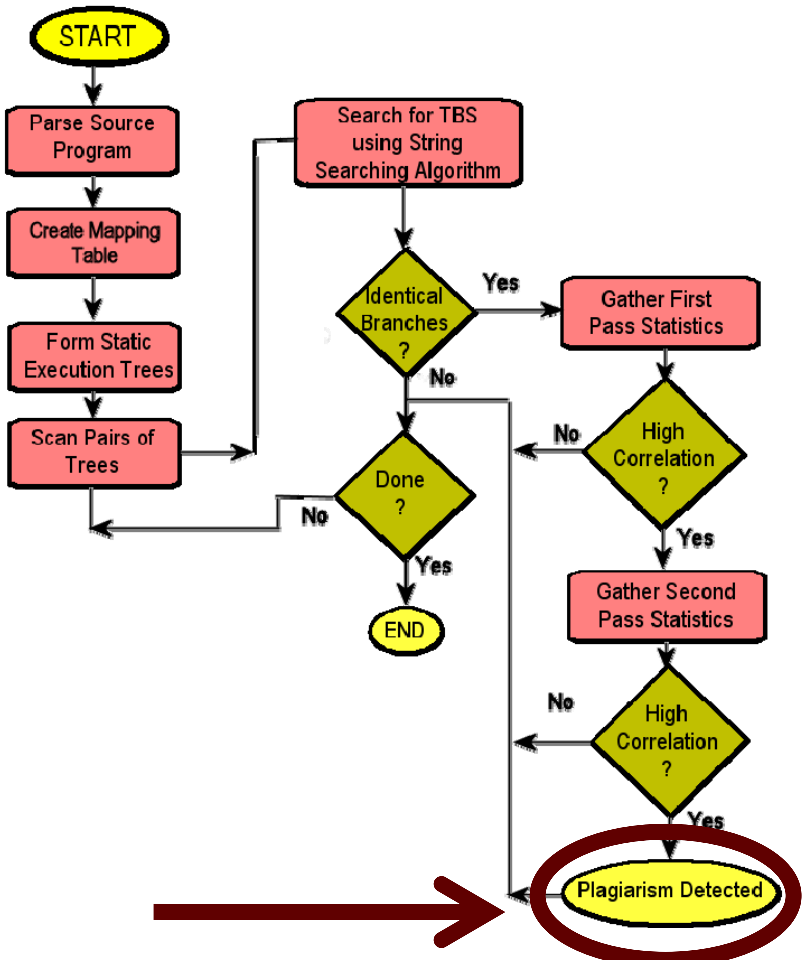
- M. H. Halstead. *Elements of Software Science*. New York: Elsevier, 1977
  - $n1$ = number of unique operators
  - $n2$ = number of unique operands
  - $N1$ = number of operator occurrences
  - $N2$ = number of operand occurrences
- $V$ = “volume” of a program
  - $V = (N1 + N2) \log2 (n1 + n2)$
- $E$ = mental effort required
  - $E = [n1\ N2(N1 + N2) \log2 (n1 + n2)]/(2n2)$

# Plagiarism Measurement

- Parker and Hamblen
  - “Computer Algorithms for Plagiarism Detection,” *IEEE Transactions on Education*, Vol. 32, No. 2, pp. 94-99, May 1989
  - Survey of various detection programs and algorithms
  - Assigned metrics to source code features
  - Specific features and number of features varied

## Plagiarism Measurement

- H. T. Jankowitz, “Detecting plagiarism in student Pascal programs,” *Computer Journal*, vol. 31, no. 1, pp. 1-8, 1988



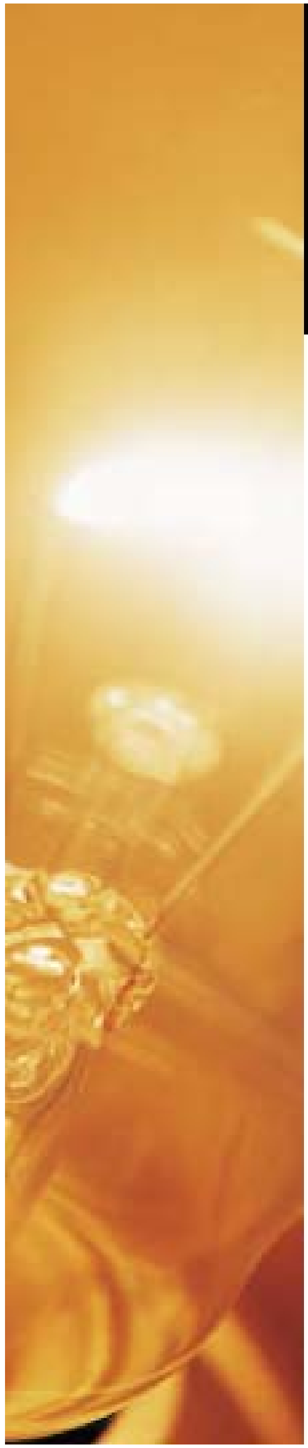
# Plagiarism Measurement

- Random House Unabridged Dictionary. (2006). Random House, Inc.

*the <u>unauthorized</u> use or close imitation of the language and thoughts of another author and the representation of them as one's own original work.*

## Plagiarism Measurement

- “Plagiarism detection”
- No definition
- No references
- No standards
- No theoretical basis
- Often reflect the creator’s bias
- Need an all-encompassing metric



## Source Code Correlation

- $\rho_s$ Statement correlation
- $\rho_c$ Comment/String correlation
- $\rho_i$ Identifier correlation
- $\rho_q$ Instruction sequence correlation
- $\rho$ Overall source code correlation

## S.A.F.E. Tests

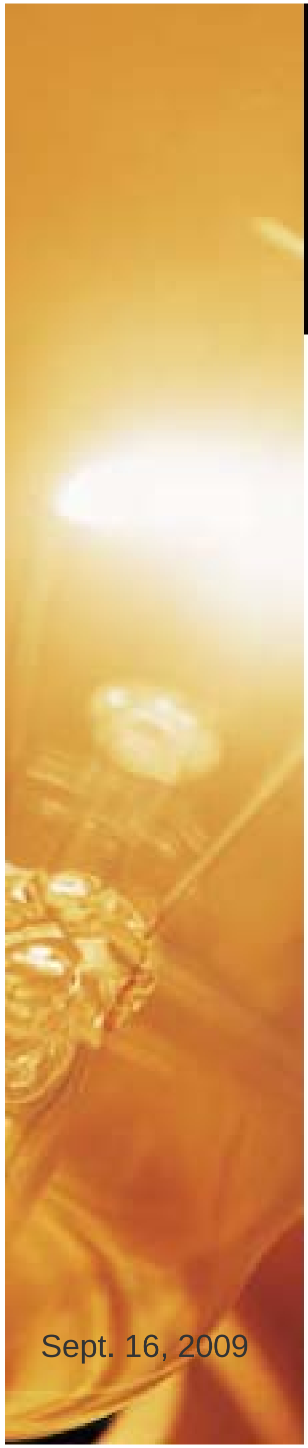
- Used C source code files from the open source GNU C compiler GCC version 3.3.2
- Arbitrarily chose ten files in each of the following categories:
  - Small: Less than 100 lines
  - Medium: Between 100 and 1000 lines
  - Large: Greater than 1000 lines

# Modifications

1. Remove comments
2. Rename identifiers
3. Rearrange routines within each file
4. Rearrange lines of code within routines
5. Do all of the above
6. Remove statements but leave comments
7. Mix selected routines into one file

# Results

| | CodeMatch | JPlag | MOSS |
|---|---|---|---|
| **Comment removed** | 100% (30 of 30) | 100% (30 of 30) | 97% (29 of 30) |
| **Identifiers renamed** | 100% (30 of 30) | 100% (30 of 30) | 97% (29 of 30) |
| **Routines rearranged** | 100% (30 of 30) | 100% (30 of 30) | 83% (25 of 30) |
| **Lines of code rearranged** | 100% (30 of 30) | 100% (30 of 30) | 87% (26 of 30) |
| **All of the above** | 100% (30 of 30) | 100% (30 of 30) | 73% (22 of 30) |
| **Code removed** | 83% (25 of 30) | 0% (0 of 30) | 0% (0 of 30) |
| **One routine from each file** | 83% (25 of 30) | 63% (19 of 30) | 50% (15 of 30) |
| **Overall** | 95% (200 of 210) | 80% (169 of 210) | 70% (146 of 210) |



## Poor tests?

- Biased toward CodeSuite®?
- Not real-life examples?
  - Academia
  - Industry
- Not independent?

## Depository of Universal Plagiarism Examples

- Choose open source projects
- Minimum definition of software plagiarism
- Logistics
  - Create database
  - Create policies
    - How to run the tests
    - How to generate the results
    - How to distribute the results
- Understand legal issues
  - Privacy
  - Copyright
  - Licensing

## Choose open source projects

- Choose one or several open source projects that include a test-bed for testing that the software is working correctly.

## Minimum definition of software plagiarism

- Must perform the same exact function as the original (must pass the test bed tests)
- Must take the same data inputs using the same data types
- Must produce the same exact outputs using the same data types

# Logistics

- Announce the Depository for Universal Plagiarism Examples  (DUPE) as an independent, unbiased, academic collection of code that will be used to test programs that detect source code plagiarism.
- Distribute the code we've chosen (or point users to the code) and request plagiarized copies to be added to DUPE.
- Offer a reward to those who contribute and those who actually fool the programs. Maybe the reward is recognition on the DUPE website.

## Logistics

- After enough entries have been received, run several plagiarism detection programs and report the results such as:
  - Which programs found which plagiarized code?
  - Percentage of false positives
  - Percentage of false negatives
- Contact providers of the plagiarism detection programs and ask them to comment and/or provide updated versions of their programs to test.

## Logistics

- Write up the results
- Continue to receive plagiarized code
- Hold comparisons regularly
- Keep the web page up to date

## Discussion/Partners

## Thank You

Bob Zeidman

bob@SAFE-corp.biz

S.A.F.E.
Software Analysis & Forensic Engineering Corp.

www.SAFE-corp.biz