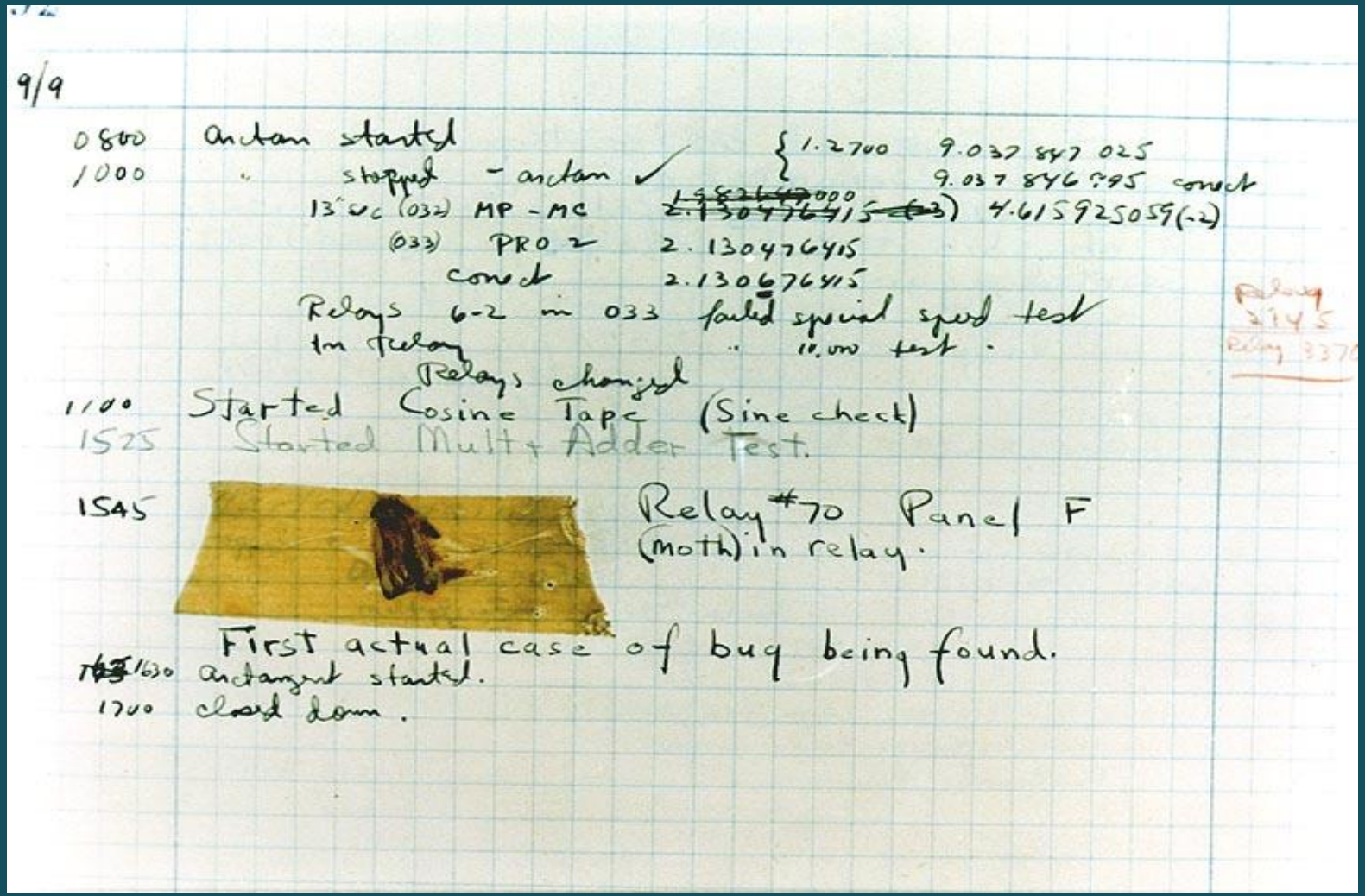
SherlockJ

Unealtă de depanarea statistică a programelor Java

Student:
Cristian-Alexandru STAICU

Coordonator științific:
Conf. dr. ing. Marius MINEA

Introducere - Istoric

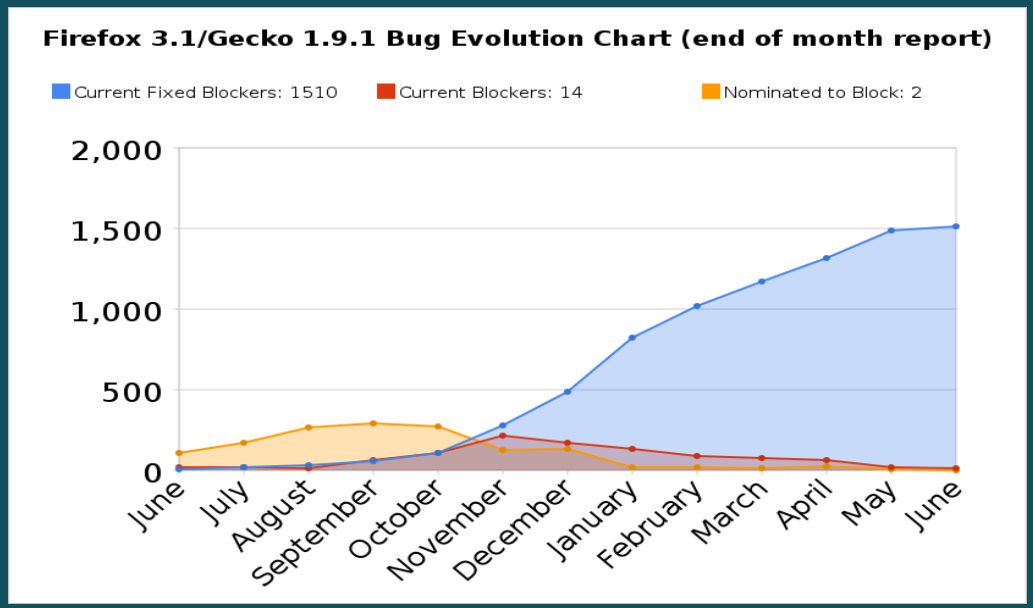


9 September 1947 - primul bug, consemnat de Grace Hopper

Introducere - Statistici

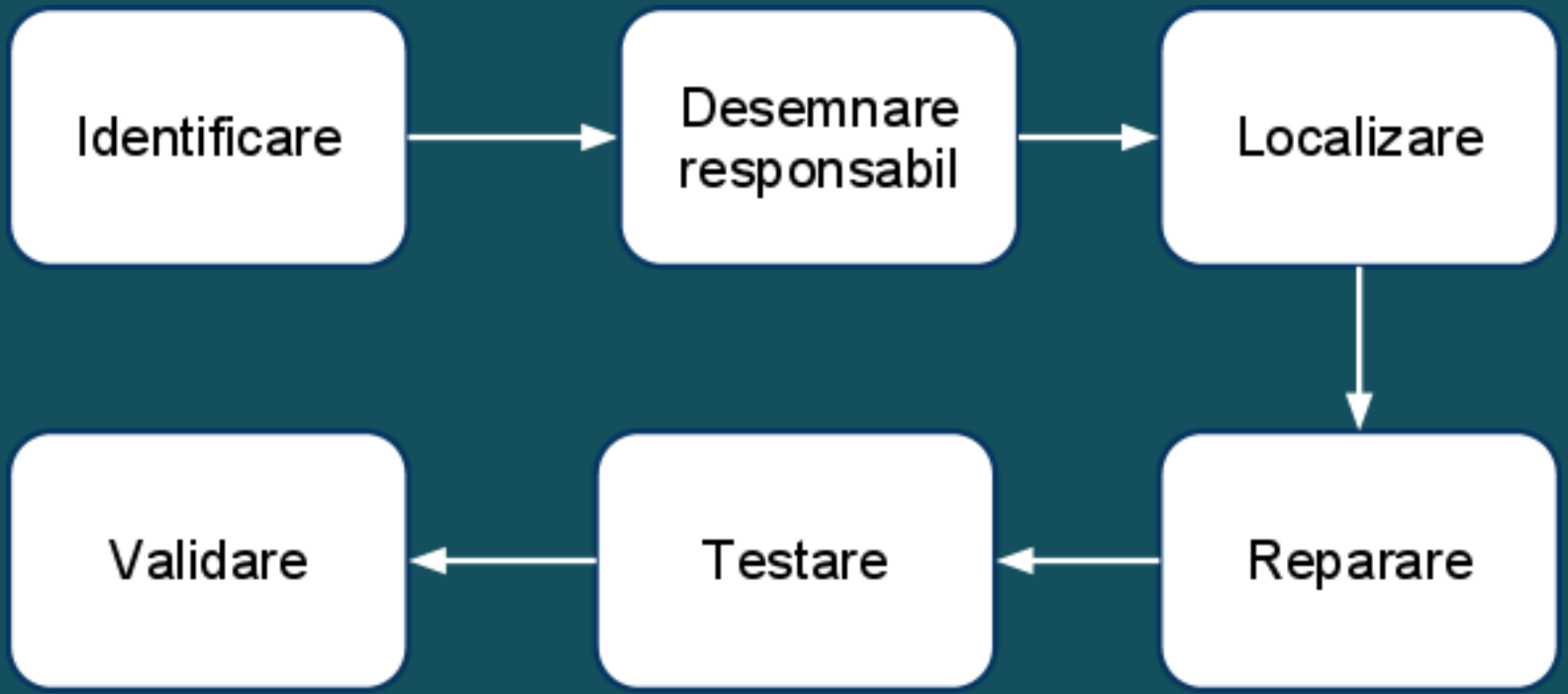
Un dezvoltator petrece în medie o treime din timp în procesul de depanare de erori.[1]

La 1000 de linii de cod există între 20 și 30 de erori.[2]

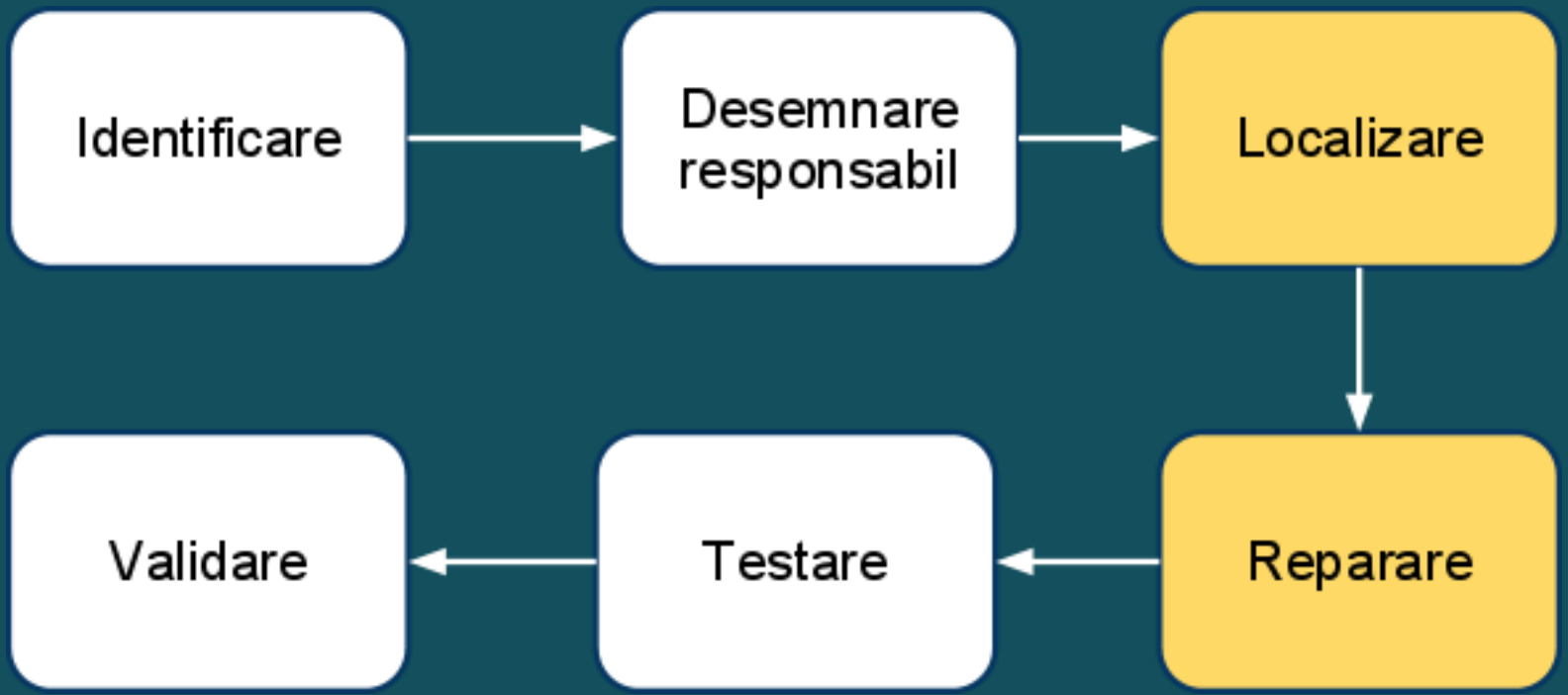


1 - studiu efectuat de IDC în 2008 pe 139 de corporații producătoare de software
2 – cf. CyLab Sustainable Computing Consortium, Carnegie Mellon University

Introducere - Ciclul de viață al unei erori



Introducere - Ciclul de viață al unei erori

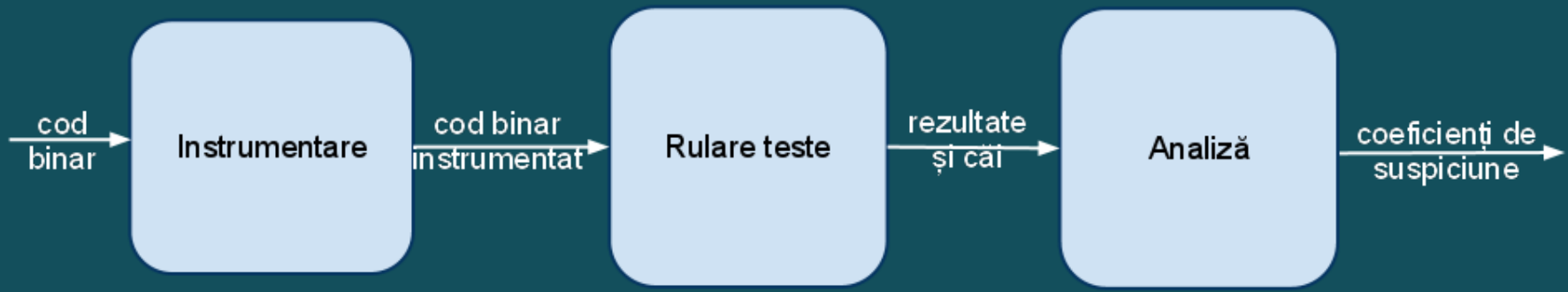


SherlockJ - Obiective

- automatizarea localizării erorilor folosind testele unitare existente
  - JUnit
  - TestNG
- ușor de folosit - integrabil în mediul de dezvoltare

- să nu altereze în vreun fel proiectul analizat
- să introducă o penalizare cât mai mică

SherlockJ - Pașii analizei



SherlockJ – Tehnici de localizare de erori

Tehnicile de depanare automată:

- asociază fiecărei entităţi din program o probabilitate ca acea entitate să conţină erori
- nu au nevoie de modele ale programului analizat

Un algoritm de localizare de erori are nevoie de:

- rezultatele rulării unor teste
- spectrul program pentru fiecare test

Tehnici de localizare de erori - exemple

Coeficientul Tarantula[1]

$$suspiciunea(e) = \frac{\frac{eronate(e)}{total\_eronate}}{\frac{corecte(e)}{total\_corecte} + \frac{eronate(e)}{total\_eronate}}$$

Coeficientul Jaccard[2]

$$suspiciunea(e) = \frac{eronate(e)}{total\_eronate + corecte(e)}$$

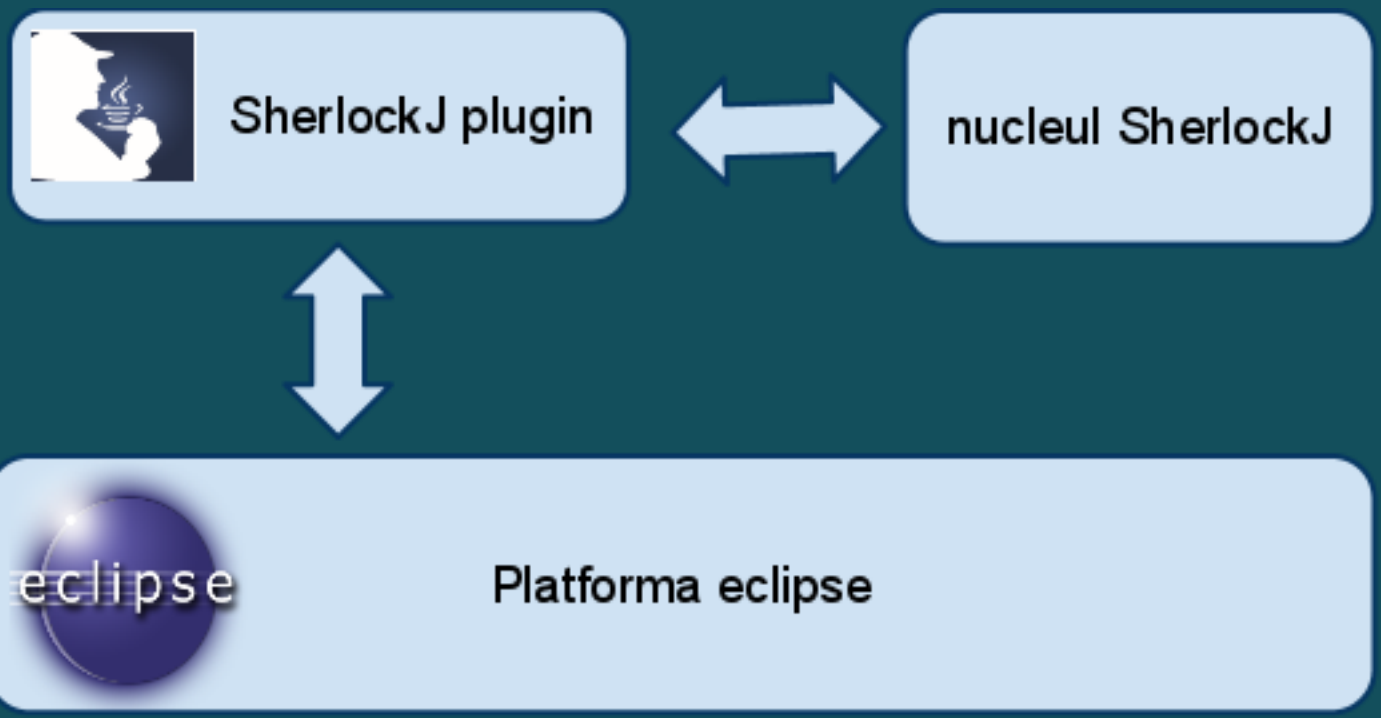
1 – Jones & Harrold, 2001
2 – Paul Jaccard, 1908

Tarantula - Exemplu

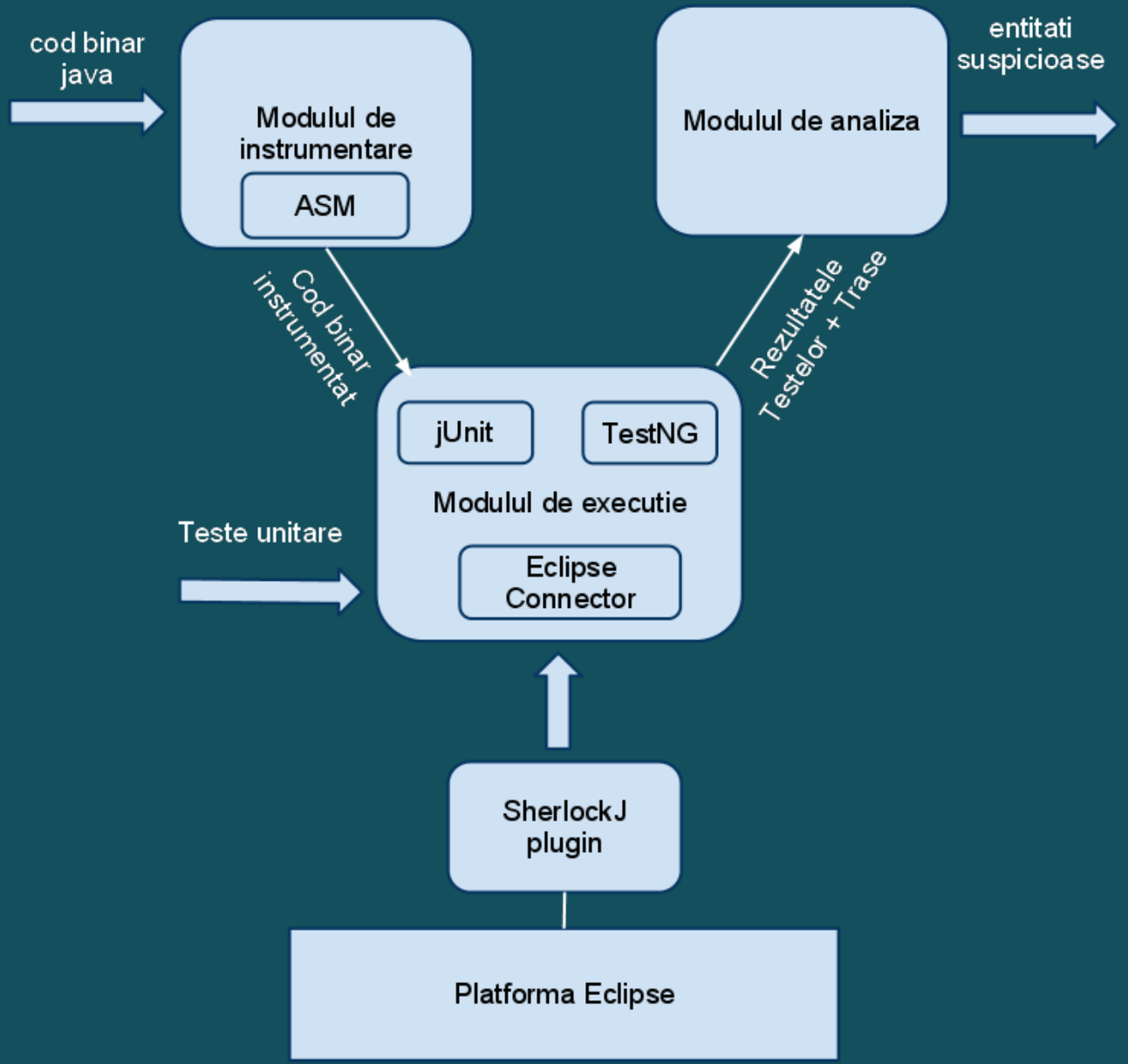
		Test Cases							
	mid() { int x,y,z,m;	3,3,5	1,2,3	3,2,1	5,5,5	5,3,4	2,1,3	suspiciousness	rank
1:	read("Enter 3 numbers:",x,y,z);	●	●	●	●	●	●	0.5	7
2:	m = z;	●	●	●	●	●	●	0.5	7
3:	if (y<z)	●	●	●	●	●	●	0.5	7
4:	if (x<y)	●	●			●	●	0.63	3
5:	m = y;		●					0.0	13
6:	else if (x<z)	●				●	●	0.71	2
7:	m = y; // *** bug ***	●					●	0.83	1
8:	else			●	●			0.0	13
9:	if (x>y)			●	●			0.0	13
10:	m = y;			●				0.0	13
11:	else if (x>z)				●			0.0	13
12:	m = x;							0.0	13
13:	print("Middle number is:",m);	●	●	●	●	●	●	0.5	7
	} Pass/Fail Status	P	P	P	P	P	F		

Jones & Harrold, 2005

SherlockJ - Arhitectura

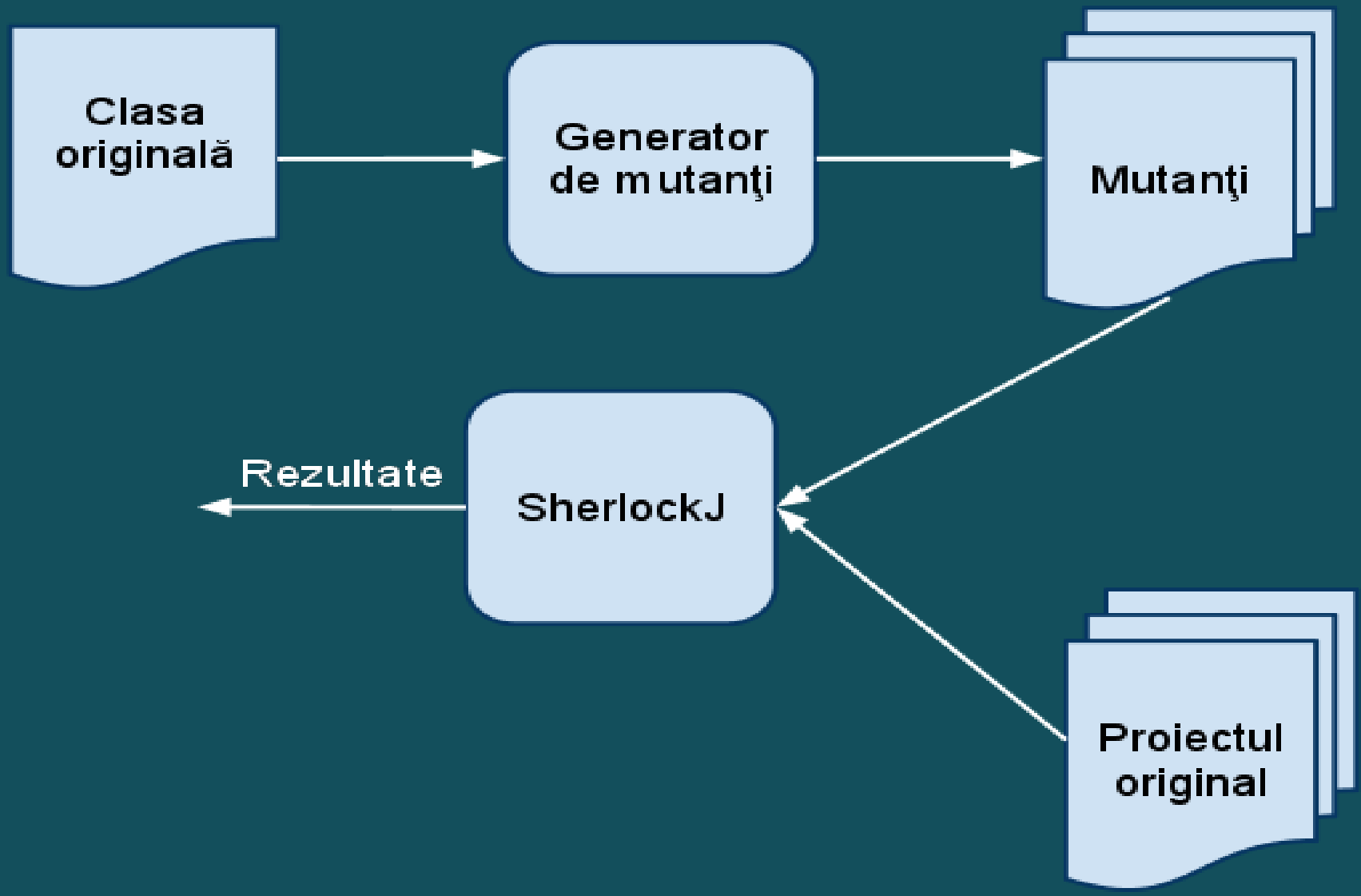


SherlockJ - Arhitectura (2)



SherlockJ - Evaluarea : Generare de mutanţi

Mutant = versiuni cu mici modificări ale claselor binare



SherlockJ – Rezultate (1)

Am analizat Google Guice:

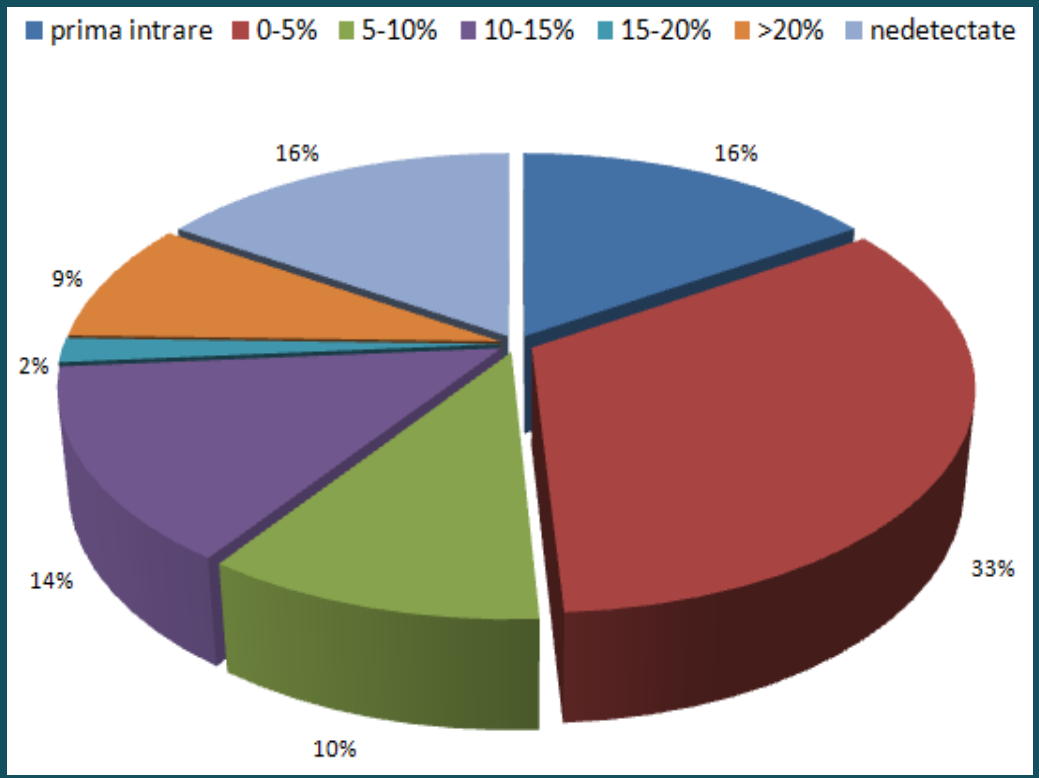
- 29.975 LOC şi 1.850 NOC
- am rulat 108 teste din pachetul com.google.inject.spi
- am generat 48 de mutanţi ai clasei InjectionPoint
- am observat acurateţea localizării erorilor din fiecare mutant

Penalizarea introdusă de instrumentare:

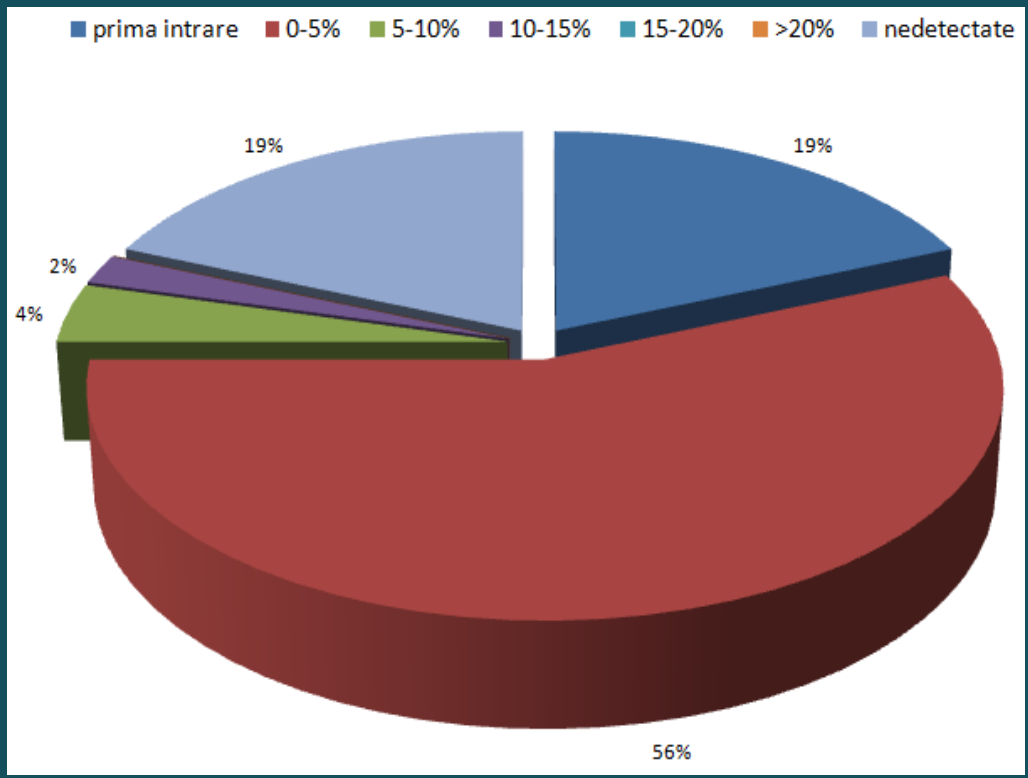
- spaţiu: 19%
- timp: 69%

SherlockJ – Rezultate (2)

Algoritmul Tarantula



Algoritmul Jaccard



Acurateţea medie de diagnoză:

- 5,97% la Tarantula
- 1.79% la Jaccard.

SherlockJ - Direcţii de dezvoltare

- Posibilitatea de modificare a granularităţii analizei
- Introducerea unei metode de vizualizare
- Rezolvarea problemelor de scalabilitate prin instrumentare selectivă

SherlockJ - Concluzii

SherlockJ este un plugin de eclipse pentru localizarea erorilor:

- extensibil
- implementează algoritmi de localizare de erori consacraţi
- uşor de folosit
- introduce un balast relativ mic
- suportă cadrele de testare consacrate JUnit, TestNG

Vă invit să îl încercaţi!
http://code.google.com/p/junit-debugger/

Vă mulțumesc pentru atenție!

Cristian-Alexandru STAICU
27 iunie 2011