



# Laboratory of Software Analysis

## *Regression Testing*

Filippo Ricca

ITC-Irst


Istituto per la ricerca  
Scientifica e Tecnologica

[ricca@itc.it](mailto:ricca@itc.it)



# The project



- ◆ **JConsole**
  - ◆ **Reverse engineering. Collect some views (Class Diagram)**
  - ◆ **Instrumentation of the JAVA code using TXL and writing test cases with statement/branch coverage.**
  - ◆ **Change requirement - add a new crosscutting feature to Jconsole**
  - ◆ **Extend the JAVA grammar for the ASPECTJ language**
  - ◆ **Instrumentation of the ASPECTJ code using TXL and writing test cases with statement/branch coverage.**
  - ◆ **Regression testing (no side effects)**
- 



# Regression Testing

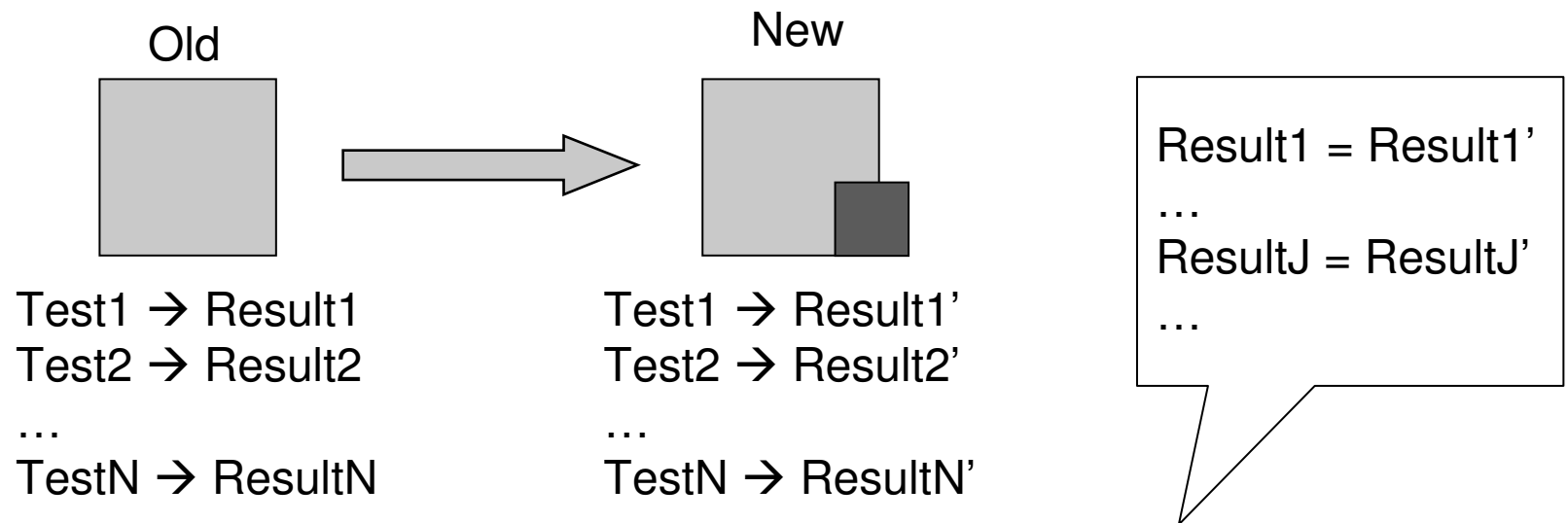


**Regression testing:** any repetition of tests (usually after software or data change) intended to show that the software's behavior is unchanged except insofar as required by the change to the software or data.

It is a quality control measure to ensure that the newly modified code still complies with its specified requirements and that unmodified code has not been affected by the maintenance activity.

# Aim of Regression testing

- ◆ **Aim:** Finding side effects (unexpected behaviors) in the new version of code.



**Results of testcase related to unchanged functionalities should be the same in the two versions of software.**



# Automatic Regression testing

- ◆ Usually, regression testing require a **large amount of effort**: the order of several weeks for medium size programs --> money and time!
- ◆ A solution to simplify it can be: Automatic Regression testing.



---

# “Partial” regression testing

---

- ◆ If the time (and money ...) interval allocated to regression testing is limited, **only a fraction of the test suite can be executed**, with the risk of missing test cases that are able to reveal defect not yet discovered.



# Techniques ...



- **[Regression test selection:]** the cost of regression testing is reduced by selecting a subset of the existing test suite based on information about the program, modified version and test suite.
- **[Test suite minimization:]** the test suite is reduced to a minimal subset that maintains the same coverage as the original test suite with respect to a given coverage criterion.
- **[Test case prioritization:]** test cases are ordered so that those with the highest priority are executed earlier, for example with the objective of achieving code coverage at the fastest possible rate, or of exercising the modules according to their propensity to fail.

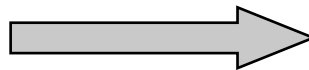
# JConsole: Regression testing

Original Jconsole



Test1 → Result1  
Test2 → Result2  
...  
TestN → ResultN

Jconsole Persistent history



Aspect

Test1 → Result1'  
Test2 → Result2'  
...  
TestN → ResultN'

*Regression*



Result1 = Result1'  
Result2 = Result2'  
...  
ResultN = ResultN'

...  
TestM → ResultM



*Testing*



# JConsole: Regression Testing

---

---

- ◆ In our case the number of testcases is not so high. For this reason we can execute “complete regression testing”.
- ◆ It’s better **Automatic Regression Testing**. To do this we need a software infrastructure.



---

# Software infrastructure

---

- ◆ The idea is realize a software module that:
  - To run testcase suite using original Jconsole
  - To run the same testcase suite using the modified Jconsole
  - To execute “diff” between Results.
- ◆ Possible differences (side effects) should be visualized in a report.