Regression testing

Davide Falessi

Regression Testing

- *Regression testing* is the process of re-testing software that has been *modified*.

- Regression testing constitutes the vast *majority* of testing *effort* in commercial software development and is an essential part of any viable software development process.

Regression Testing

- Even though many developers don't want to believe it (even when faced with indisputable evidence!), *small changes to one part of a system often cause problems in other parts of the system.*
- Regression testing is used to find this kind of problem.

You cannot test everything

- Test set cannot be run *as often as changes* are made to the software.
- *Regression tests run at night* to evaluate software changed that day, with developers reviewing the results the following morning.
- If the regression tests do not finish in a *timely manner*, the development process is disrupted.

Again, you cannot test everything

- At some point, *the marginal advantage of adding a given test is not worth the marginal expenditure* of the resources needed to execute it.
- On the other side, a set that is too small will not cover the functionality of the software sufficiently well, and too many faults will make it past the regression test set to the customer.

How to reduce regression testing

- Limiting the amount of time needed to execute regression tests, and a focus of much of the attention in the research literature, is *selecting* only a subset of the regression tests.
  - E.g.: If the execution of a given test case does not visit *anything* modified, then the test case has to perform the same both before and after the modification, and hence can be safely omitted.

Types of changes

- Changes to software are often classified as:
  - **Adaptive**: modifying the system to cope with changes in the software environment (e.g., a new OS).
  - **Corrective**: diagnosing and fixing errors, especially the ones found by users (e.g., fixing a bug).
  - **Perfective**: improving the system performances (e.g., improving the UI from B/W to color).
  - **Preventive**: increasing software maintainability or reliability to prevent problems in the future (e.g., changing type of DB to accommodate more users).