Strategy
for software quality improvement

before you introduce unit-testing

Two „simple“ problems:

1) How many test-cases could you define for this problem?

The program reads three integer values from an input dialog. The three values represent the lengths of the sides of a triangle. The program displays a message that states whether the triangle is scalene, isosceles, or equilateral.

Glenford J. Myers, The Art of Software Testing

2) How many execution paths could there be in the following code?

```
String EvaluateSalaryAndReturnName( Employee e )
{
  if( e.Title() == "CEO" || e.Salary() > 100000 )
  {
    cout << e.First() << " " << e.Last() << " is overpaid" << endl;
  }
  return e.First() + " " + e.Last();
}
```

Herb Sutter, Exceptional C++, Item 18

Two „simple“ problems - the answers

1) 14

2) 23

80% of time is spent for bug-fixing
Steve McConnell***, Project Management Survival Guide***

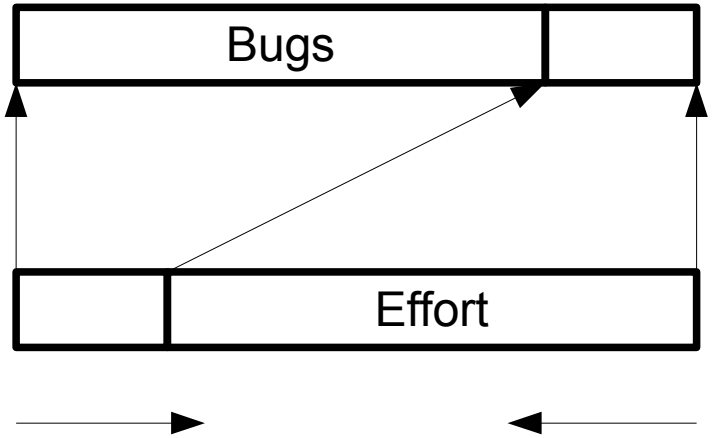
50% of time and 50% of cost are allocated to testing
Glenford J. Myers, The Art of Software Testing

Can you afford it?

80/20 Rule

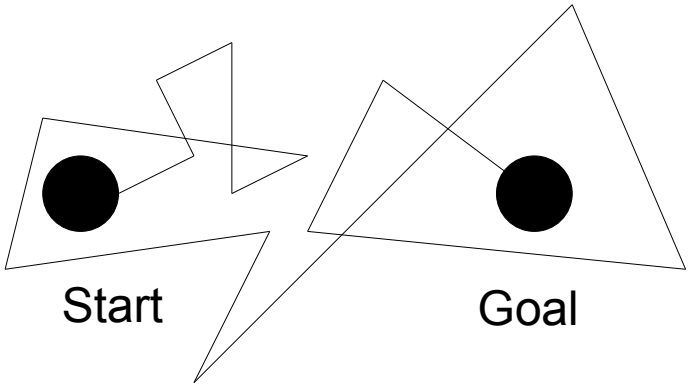
80/20 Rule is unversal. It aplies also to bugfixing:

„80% of all bugs can be found with 20% of effort,
however, for remaining 20% of bugs 80% of effort is required„

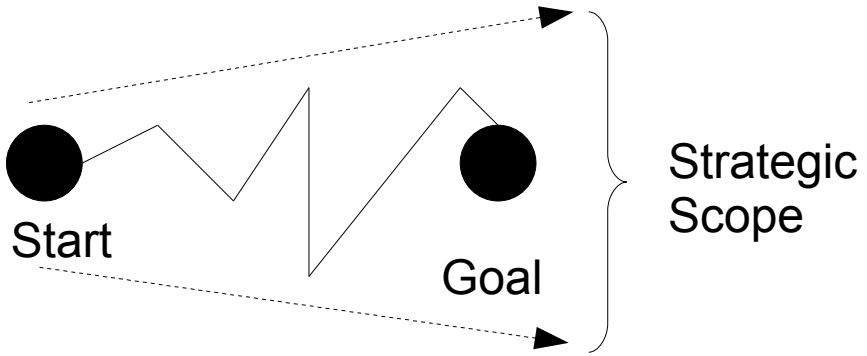


From which side would you begin?

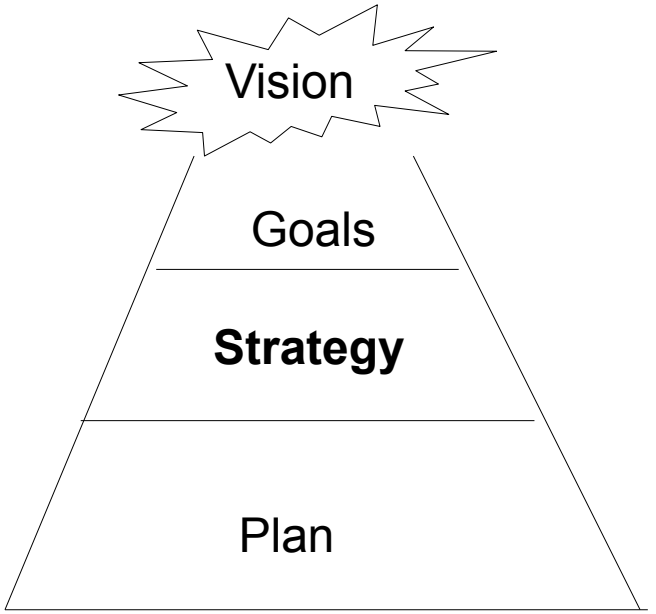
Strategy – structured, formalized way for reaching goals



a) No strategy



b) Using strategy



Strategy proposal:

- Apply best practices from software industry
- **Begin with those, which require less effort**

  - Use assertions for testing pre/post conditions
  - „No warnings in submited code“-Policy
  - Introduce formal Code Reviews
  - Start collecting Test-Cases (no implementation yet)

Assertions

```
#include <cassert>

void setMonth( int month ) {
      assert( (month > 0) && (month <= 12) );
      _month = month;
}
```

Standard assertion macro.
Execution breaks in debug-mode, if condition is true.

```
void setMonth( int month ) {
      Q_ASSERT( (month > 0) && (month <= 12) );
      _month = month;
}
```

Qt-Assertion.
Writes message using qFatal, if condition is true.

Effort: 0

Advantages:
- The whole development team is testing permanently
- Compact documentation of pre-/postconditions

„No warnings in submitted code“-policy

„If you don't have time to do it properly now, where do you want to take time for reengineering it later?“
--Steve McConnell, Software Project Survival Guide

- Enable all warning levels provided by your compiler
- Handle warnings as errors
- Check, which other code checking possibilities are offered for free by your development environment

Effort: 0

Advantages:
- Avoiding bugs instead of fixing them

Formal Code Reviews

„attac problems not people“

- Define minimalistic formal code-review process
- Agree on code style guide.
  Ready to use style guides can be found in Internet.
- Let the reviewer explain you your code

Effort: 5-15 min per submit

Advantages:
- Clean, maintainable code
- Developers share their knowledge

Start Collecting Test-Cases

„There is no point in being exact if you don't know what you are talking about“

--John von Neumann

- A minimalistic test case description consists of:
  - Testcase Id (e.g. TC_INIT_0001)
  - What should be tested (e.g. MyLib.dll init-process)
  - Expected results (e.g. ErrorCode: OK)
- Store test-case proposals in a public repository

Effort: 5 min per test-case

Advantages:
- Testcase comprehansive for all, not only for developer
- Basis for project lead to make decision, which test-cases to implement
- „Think more, play less“ - reduces risk for waisted work.