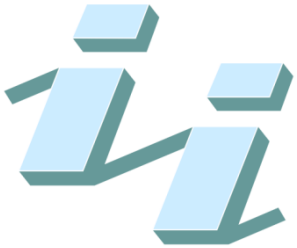
Mutation Testing of ASP.NET MVC

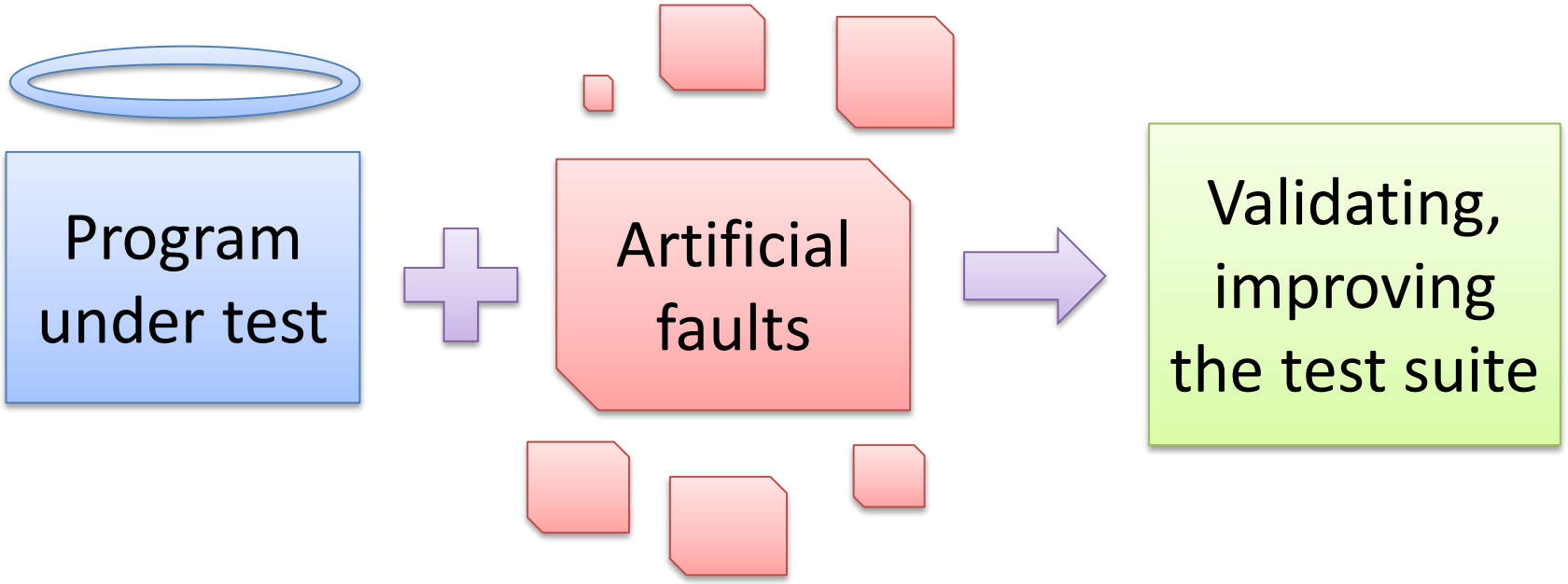


Anna Derezińska, Piotr Trzpil

Institute of Computer Science
Warsaw University of Technology

Mutation testing

Testing the tests, not the program



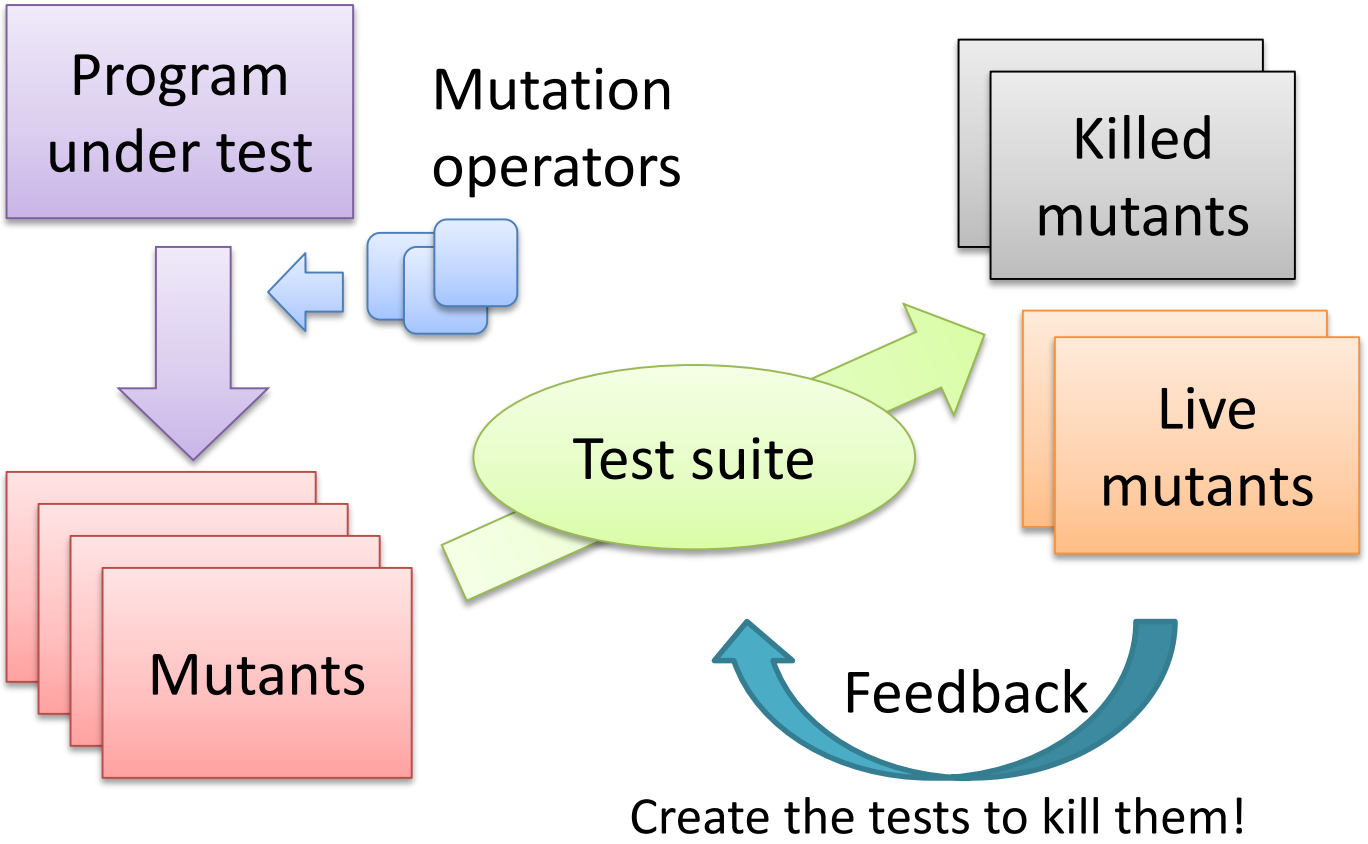
…in the Web?

Outline

1. About mutation testing process
2. ASP.NET MVC framework basics
3. Mutation testing targeted on ASP.NET MVC
4. Experiments and results
5. Future work

Mutation testing

Injecting faults into program under test in order to test and improve the test suite.



Mutation operators

Blueprints for mutants

They should be:

- Injecting simple faults
- Imitating programmer's mistakes
- Changing the meaning of the program (non-equivalent)

Normally:
one change – one mutant

Original program:

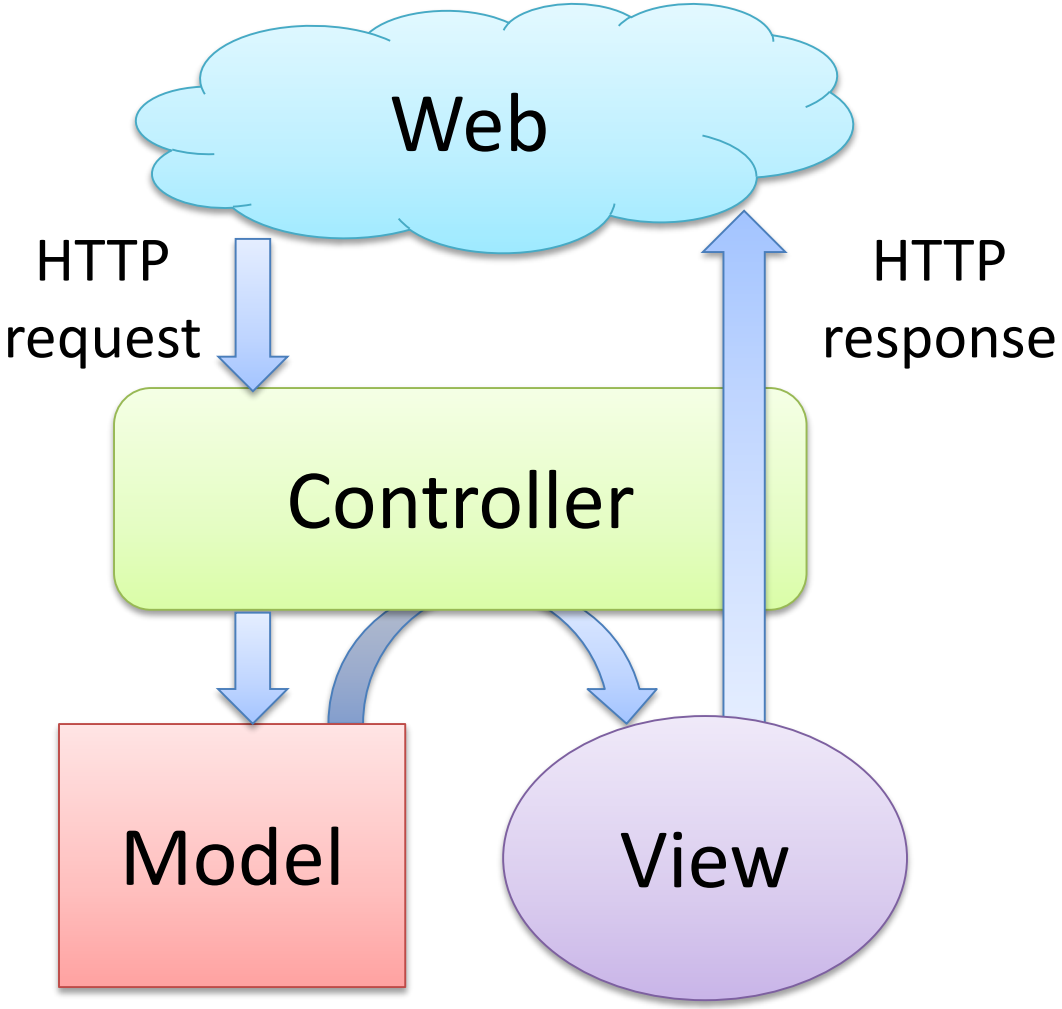
```
List<int> ChooseGreaterThan(int[] input, int val)
{
    List<int> result = new List<int>();
    foreach (int current in input)
    {
        if (current > val) result.Add(val);
    }
    return result;
}
```

Mutant:

```
List<int> ChooseGreaterThan(int[] input, int val)
{
    List<int> result = new List<int>();
    foreach (int current in input)
    {
        if (current >= val) result.Add(val);
    }
    return result;
}
```

(ROR – Relational Operator Replacement)

ASP.NET MVC framework



Controller methods

```
public class DinnersController : Controller {
  public ActionResult Index(string q)
  {
    IQueryable<Dinner> dinners =
       dinnerRepository.FindUpcomingDinners();
    return View(dinners.ToPagedList(1, PageSize));
  }

  [Authorize]
  public ActionResult Edit(int id)
  {
    Dinner dinner = dinnerRepository.Find(id);
    if (dinner == null)
      return View("NotFound");
    if (!dinner.IsHostedBy(User.Identity.Name))
      return View("InvalidOwner");
    return View(dinner);
  }
  ...
}
```

- Separated from server execution engine
- Separated from data access and view logic
- Easy to unit test

New mutation operators

RVRA - Replace View with RedirectToAction

```
return View(dinner);
```

```
return RedirectToAction("Index");
```

RAAT - Remove Authorize Attribute

```
[Authorize]
public ActionResult Edit(int id)
{
```

```
public ActionResult Edit(int id)
{
```

RAAT operator

RAAT - Remove Authorize Attribute

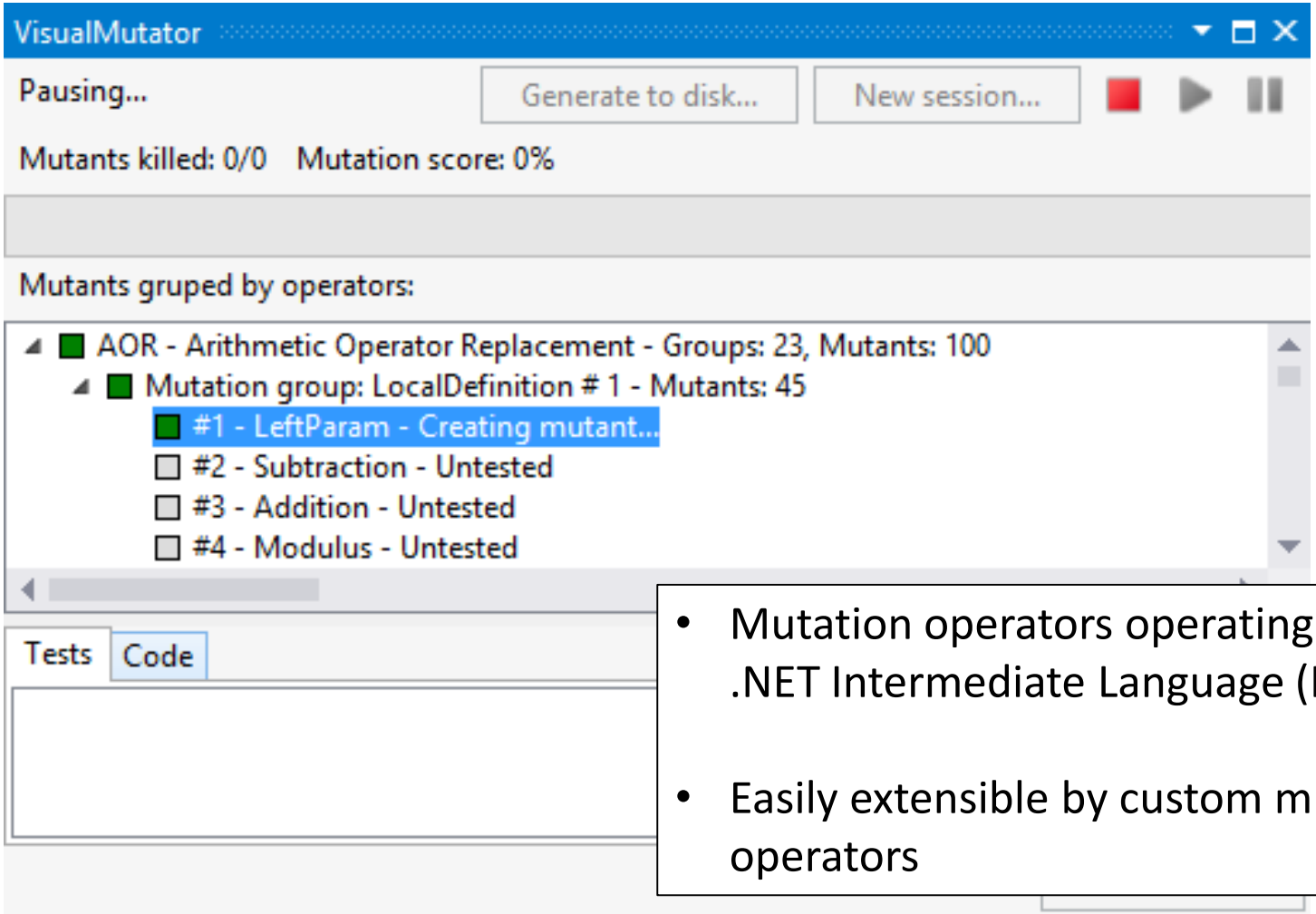
```
[Authorize]
public ActionResult Edit(int id)
{
```

```
public ActionResult Edit(int id)
{
```

- Mutant can be considered equivalent on unit test level
- …while it is non-equivalent on functional test level

VisualMutator

A Microsoft Visual Studio extension

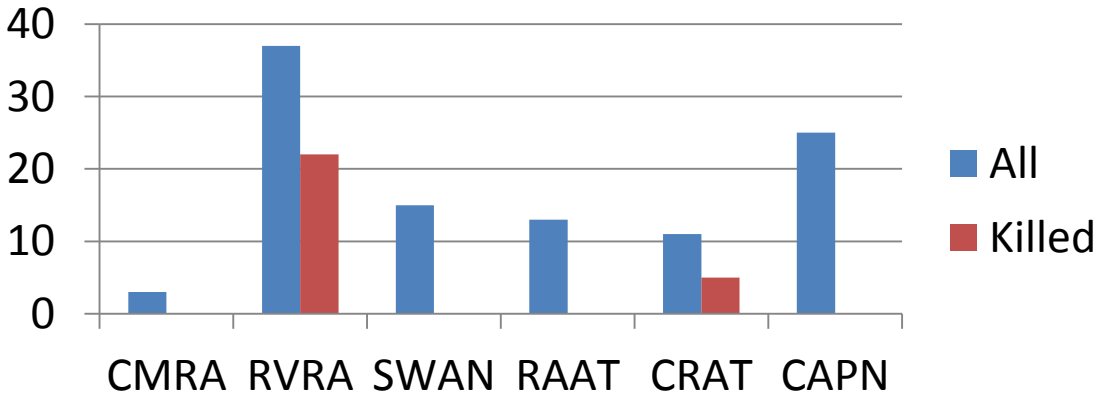


- Mutation operators operating on .NET Intermediate Language (IL)
- Easily extensible by custom mutation operators

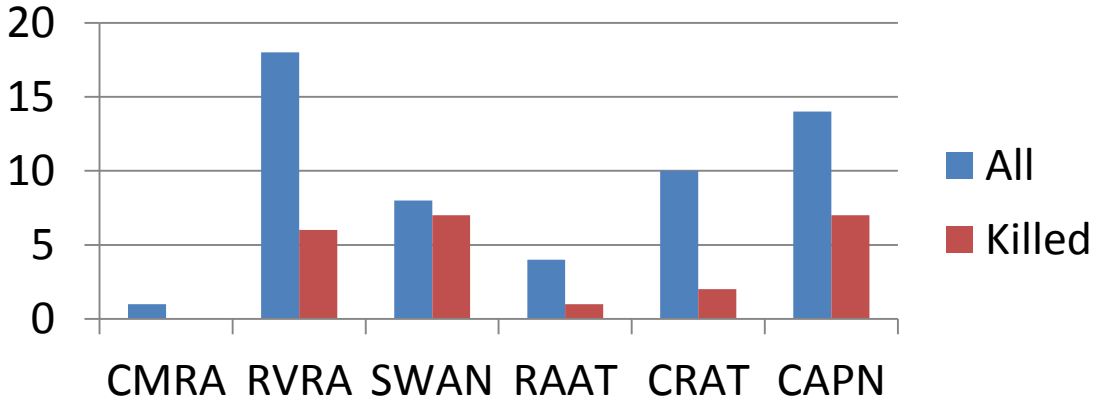
Results of experiments

Number of mutants per mutation operator

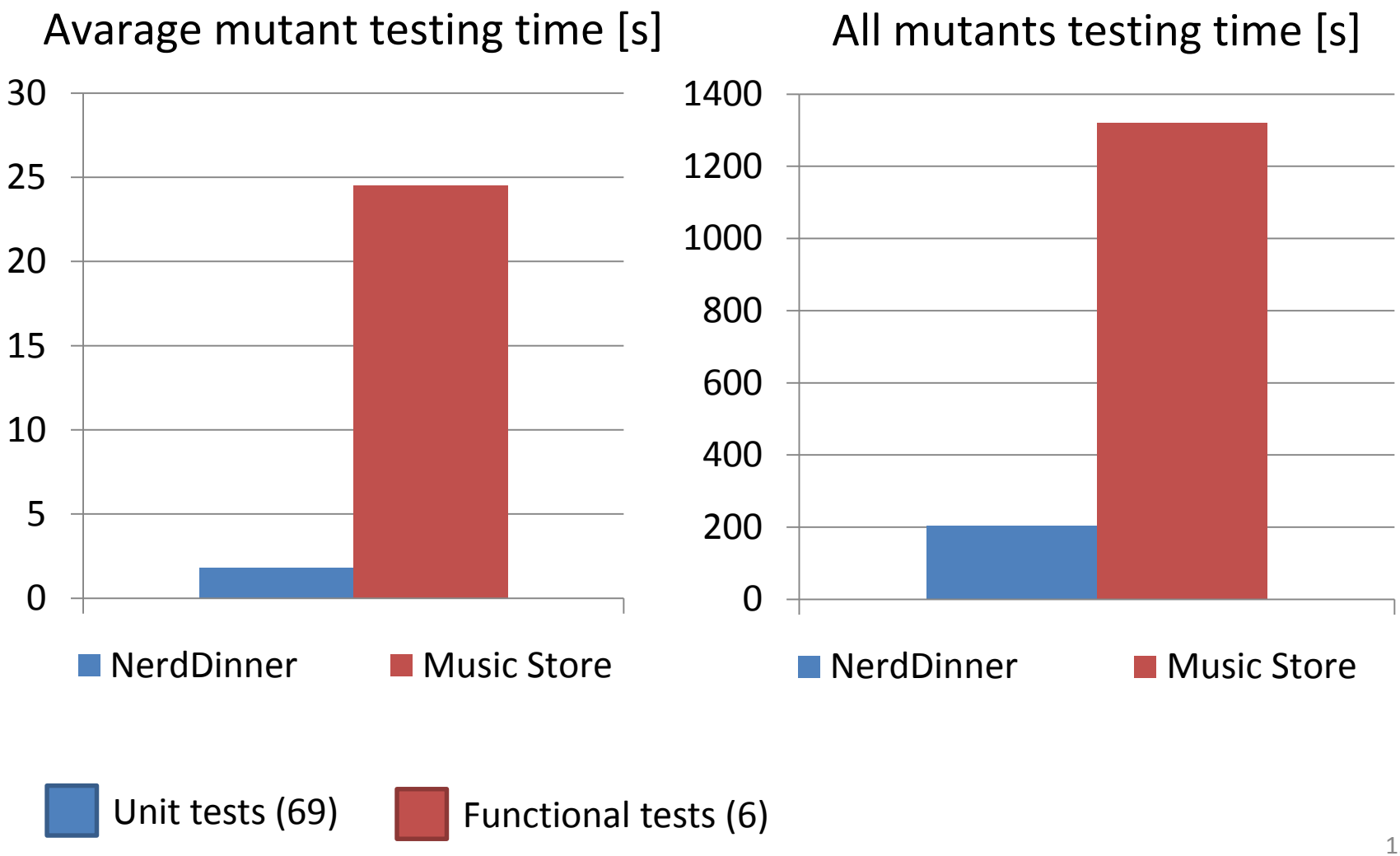
Project 1
NerdDinner:
LOC: 730
Unit tests count: 69



Project 2
MVC Music Store:
LOC: 195
Functional tests count: 6



Running time – all tests



Conclusions

- Are tests using metaprogramming a considerable choice in mutation testing?

  → Mutant equivalency becomes blurred

- Is mutation testing suited towards functional tests?

  → Long running time is a large obstacle

- Do mutation operators specific for ASP.NET MVC have a future?

  → Their results can be similar to that of some general purpose operators

VisualMutator future

- More usability improvements
- Easier extension (new mutation operators)
- Automatic unit test generation based on mutants

Thank you.