Object-oriented mutation applied in Common Intermediate Language programs originated from C#

Anna Derezińska, Karol Kowalski

Institute of Computer Science
Warsaw University of Technology

www.ii.pw.edu.pl/~adr/

Outline

- Object-oriented mutations
- Common Intermediate Language
- O-O mutations on CIL level
- ILMutator system
- Experiments
- Conclusions

Object-oriented mutations

- OO – misusing of class and object interrelations
- Locally interpreted or distributed over a whole program, e.g. class hierarchy
- Single instruction at high-level language
- Several instructions at low-level language, e.g. Common Intermediate Language
- Advanced operators – more language-related than standard (traditional) mutation operators

Advanced operators for C#

C# 1.1 Specified 40 mutation operators including:

-analogous to Java adopted for C#

- with different specifications
- with different application scope

-for specific features of C#:

delegates, properties, indexers, override modifier

C# 2.0, 3.0,.. many new features not suitable for mutation: sealed modifier, generics, partial classes and methods, extension, anonymous methods, .. (Many not applicable in the CIL)

Mutation tools for C#

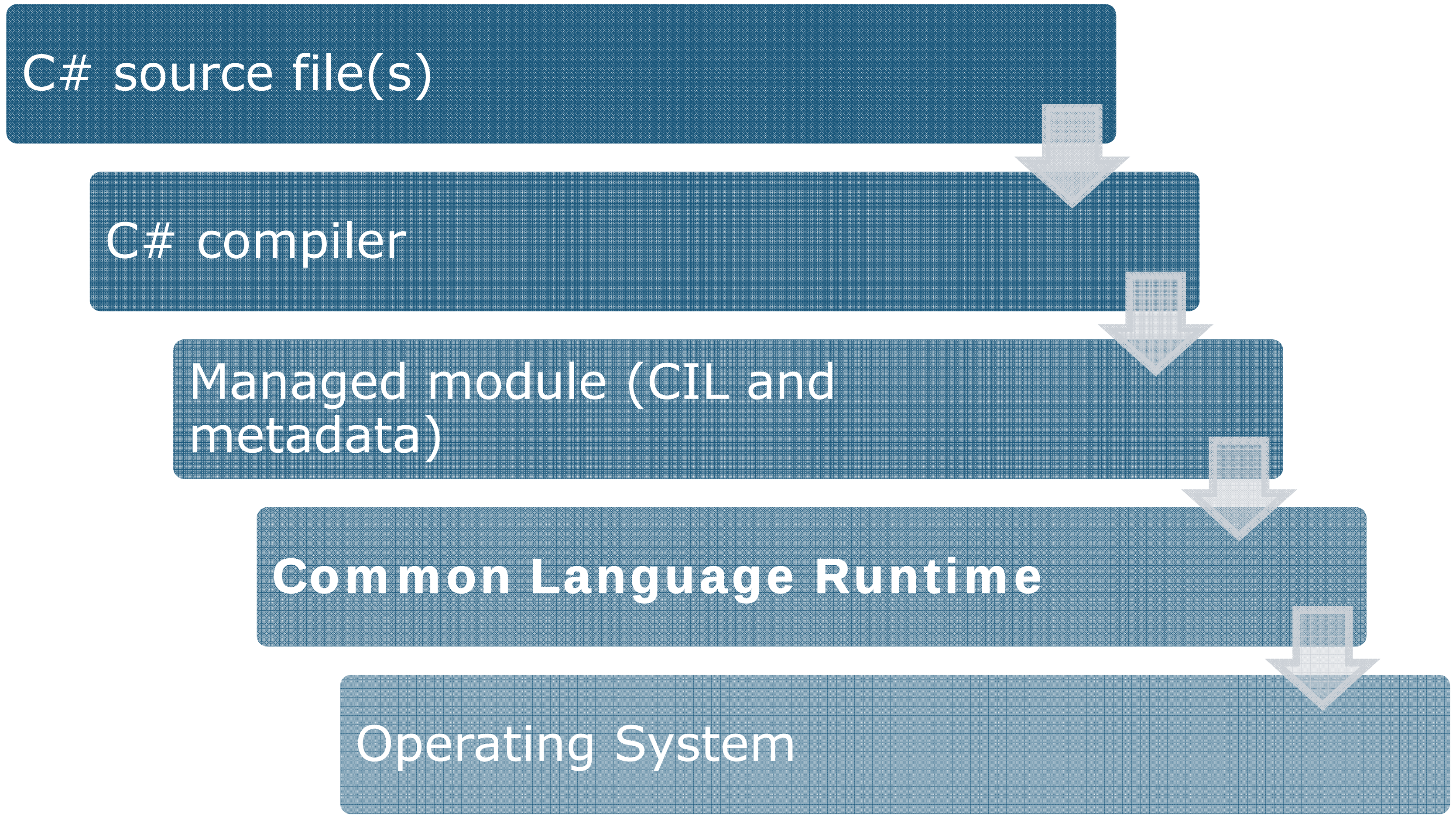
- Nester – simple mutation by use of regular expressions
- PexMutator – standard mutation operators
- CREAM – parser based,
  18 object-oriented, 8 standard mutation operators (v3)
- ILMutator – mutation operators in the Intermediate Language of .NET originated from C# code

Common Intermediate Language

- Common Language Runtime (CLR) – runtime environment of Microsoft .NET Framework
- Assembly = metadata + managed code
- Managed code = Common Intermediate Language (CIL)

- Machine level language exploiting all capabilities of CLR
- Programs translated from C# use only subset of these capabilities

Common Language Runtime

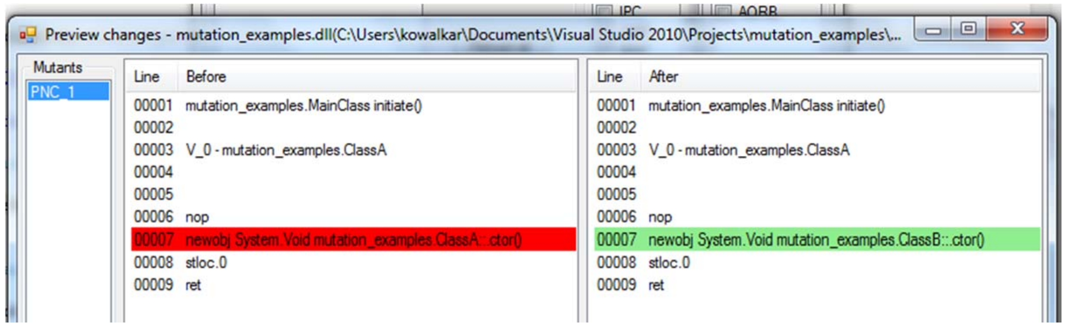


O-O mutations on CIL level

PNC – new method call with child class type

```
//C# before mutation
public class ClassA
{ }

public class ClassB : ClassA
{ }

public class MainClass
{
    static void initiate()
    {
        ClassA a;
        a = new ClassA();
    }
}
```

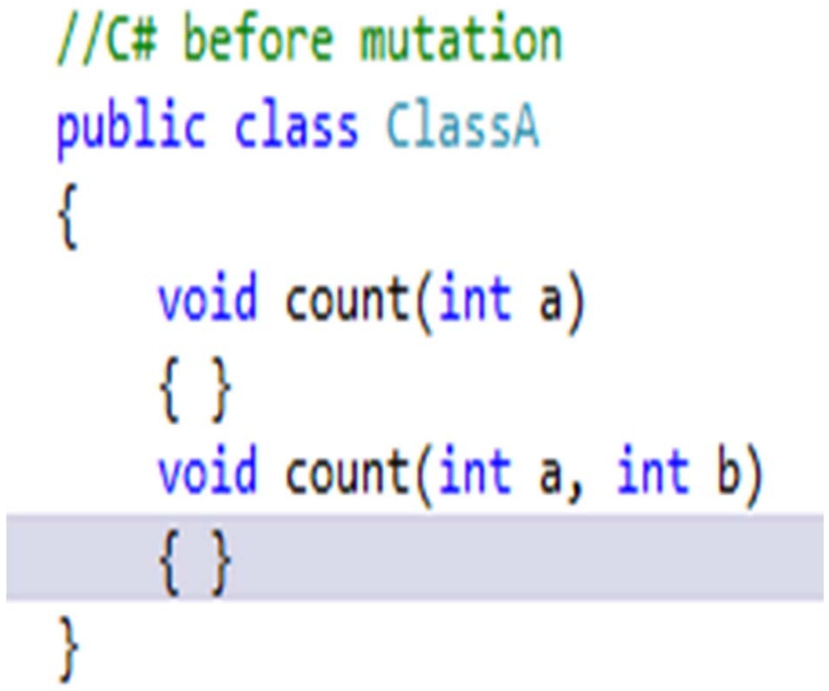
```
//C# after mutation
public class ClassA
{ }

public class ClassB : ClassA
{ }

public class MainClass
{
    static void initiate()
    {
        ClassA a;
        a = new ClassB();
    }
}
```


Preview changes - mutation_examples.dll(C:\Users\kowalkar\Documents\Visual Studio 2010\Projects\mutation_examples\...

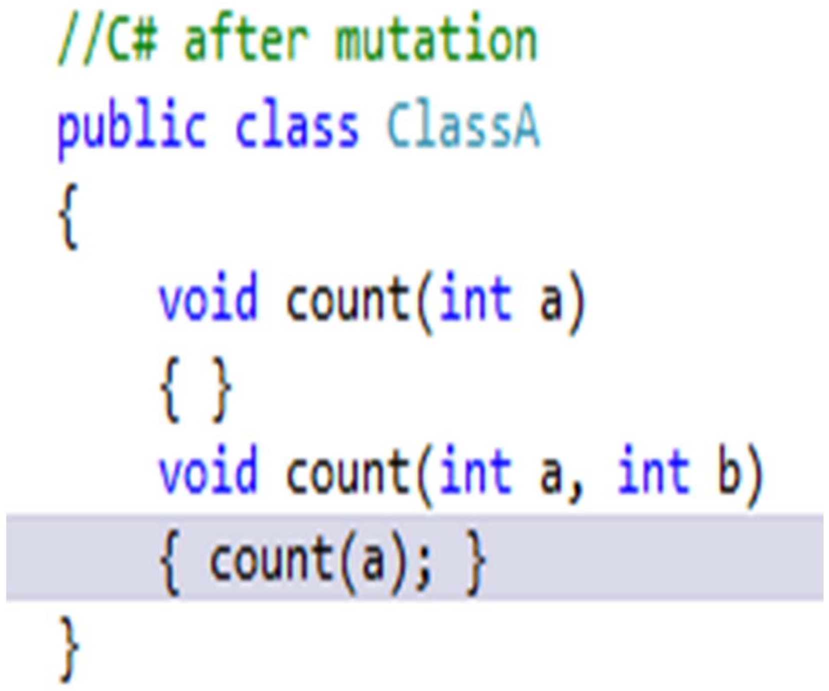
Line	Before
00001	mutation_examples.MainClass initiate()
00002	
00003	V_0 - mutation_examples.ClassA
00004	
00005	
00006	nop
00007	newobj System.Void mutation_examples.ClassA::.ctor()
00008	stloc.0
00009	ret

Line	After
00001	mutation_examples.MainClass initiate()
00002	
00003	V_0 - mutation_examples.ClassA
00004	
00005	
00006	nop
00007	newobj System.Void mutation_examples.ClassB::.ctor()
00008	stloc.0
00009	ret

OMR operator *(Overloading method contents change)*



```
//C# before mutation
public class ClassA
{
    void count(int a)
    { }
    void count(int a, int b)
    { }
}
```



```
//C# after mutation
public class ClassA
{
    void count(int a)
    { }
    void count(int a, int b)
    { count(a); }
}
```

Pre: - Avoiding recursive call of methods
- At least one consistent combination of parameters

OMR operator – in CIL

```
                    .method private hidebysig instance
                  void  count(int32 a, int32 b) cil managed
{                                    {
 .maxstack  8                         .maxstack  8
 IL_0000: nop                          IL_0000: nop
 IL_0001: ret                          IL_0001: ldarg.0
}                                      IL_0002: ldarg.1
                                       IL_0003: call instance void
                                       Operators.ClassA::count(int32)
                                       IL_0008: nop
                                       IL_0009:  ret
                                     }
```

Class constructor - 3 sections in CIL

```
//C#
 public class ClassB
    {
        private int a;
        private int b = 1;

        public ClassB()
        {
            a = 2;
          }
    }
```

```
....
 .ctor()  ....
{
// initialization of fields defined
in ClassB
e.g. a=0; b=1;
//constructor of the base class
or another constructor of this
class is called
//constructor body
e.g. a = 2;
}
```

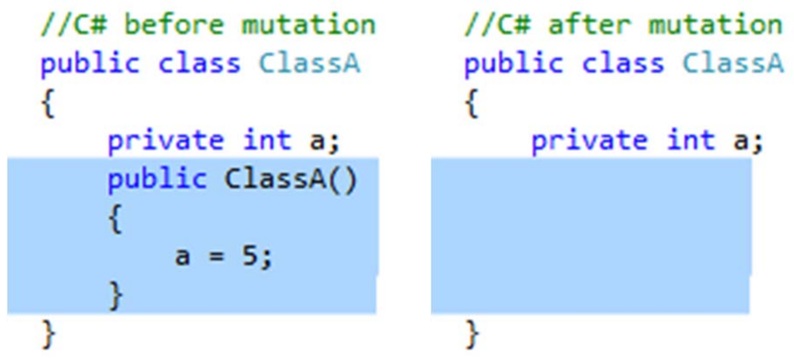
Constructors changed by operators

JDC – *C#-supported default constructor create*
Pre: A non-parametric constructor is the only class constructor
This constructor is deleted
CIL – 3rd section of the constructor is deleted
(= constructor without its body)

JID – *member variable initialization deletion*
`private int a =5;` $\longrightarrow$ `private int a;`
Initializations deleted from the 1st section of all constructors
Restriction: only primitive types

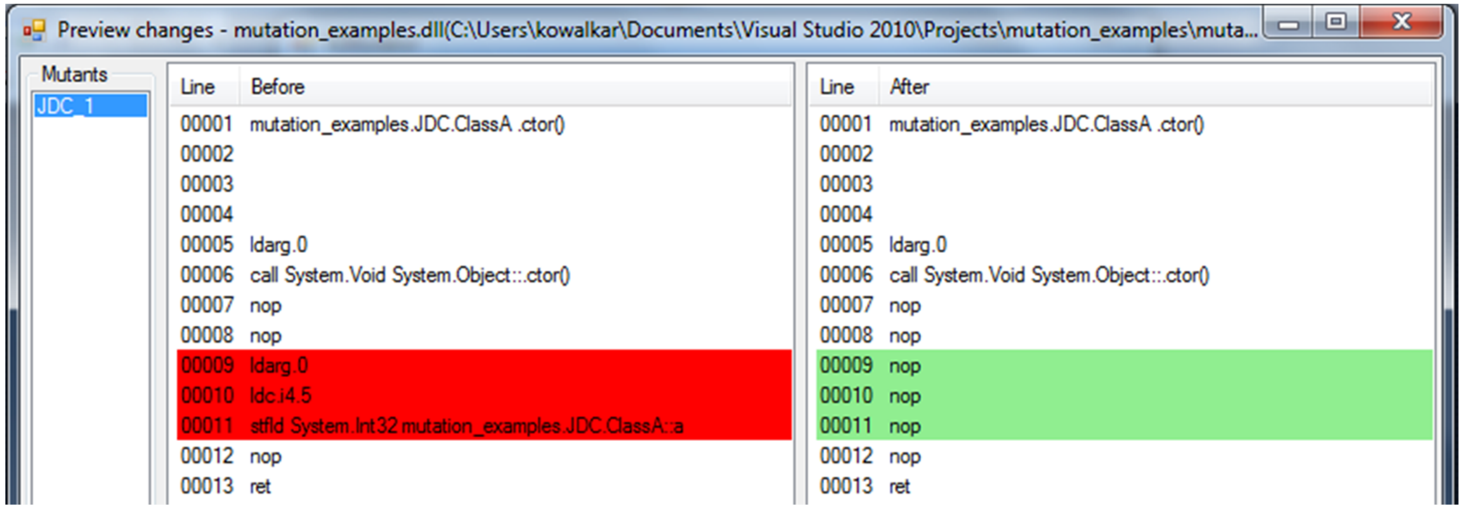
O-O mutations on CIL level

JDC – C#-supported default constructor create



```
//C# before mutation
public class ClassA
{
    private int a;
    public ClassA()
    {
        a = 5;
    }
}
```

```
//C# after mutation
public class ClassA
{
    private int a;



}
```



Preview changes - mutation_examples.dll(C:\Users\kowalkar\Documents\Visual Studio 2010\Projects\mutation_examples\muta...

Mutants: JDC_1

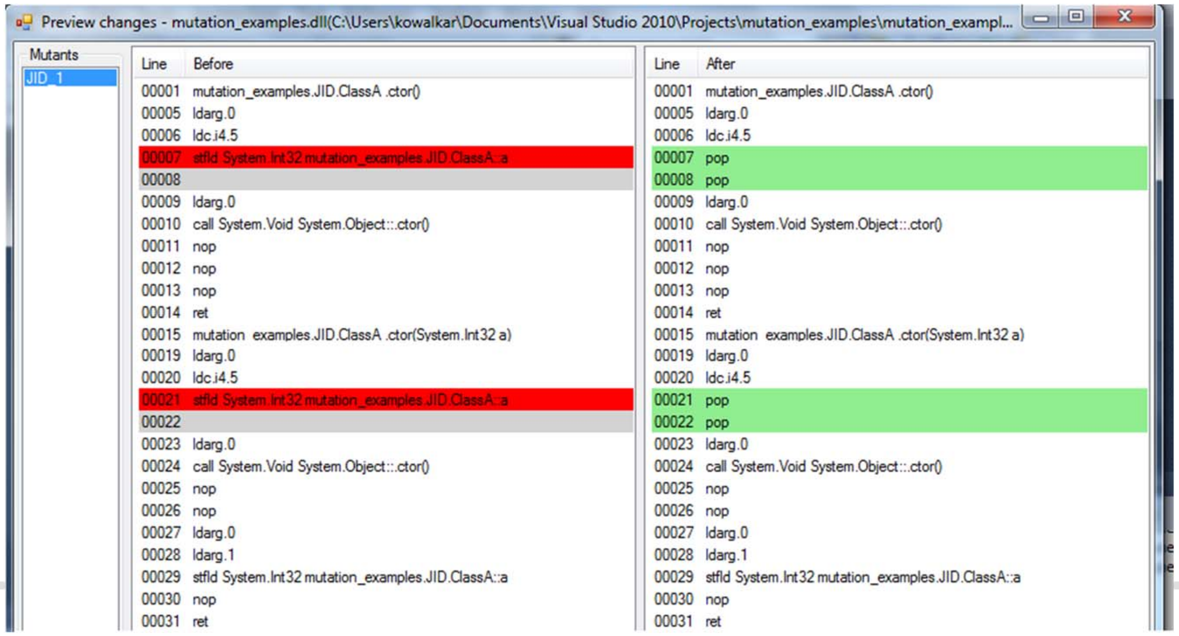
```
Line   Before                                              Line   After
00001  mutation_examples.JDC.ClassA .ctor()                00001  mutation_examples.JDC.ClassA .ctor()
00002                                                      00002
00003                                                      00003
00004                                                      00004
00005  ldarg.0                                             00005  ldarg.0
00006  call System.Void System.Object::.ctor()             00006  call System.Void System.Object::.ctor()
00007  nop                                                 00007  nop
00008  nop                                                 00008  nop
00009  ldarg.0                                             00009  nop
00010  ldc.i4.5                                            00010  nop
00011  stfld System.Int32 mutation_examples.JDC.ClassA::a  00011  nop
00012  nop                                                 00012  nop
00013  ret                                                 00013  ret
```

O-O mutations on CIL level

JID – member variable initialization deletion

```
//C# before mutation
public class ClassA
{
    private int a = 5;

    public ClassA() { }
    public ClassA(int a)
    {
        this.a = a;
    }
}
```

```
//C# after mutation
public class ClassA
{
    private int a;

    public ClassA() { }
    public ClassA(int a)
    {
        this.a = a;
    }
}
```



Preview changes - mutation_examples.dll(C:\Users\kowalkar\Documents\Visual Studio 2010\Projects\mutation_examples\mutation_exampl...

Mutants

JID_1

Line	Before	Line	After
00001	mutation_examples.JID.ClassA .ctor()	00001	mutation_examples.JID.ClassA .ctor()
00005	ldarg.0	00005	ldarg.0
00006	ldc.i4.5	00006	ldc.i4.5
00007	stfld System.Int32 mutation_examples.JID.ClassA::a	00007	pop
00008		00008	pop
00009	ldarg.0	00009	ldarg.0
00010	call System.Void System.Object::.ctor()	00010	call System.Void System.Object::.ctor()
00011	nop	00011	nop
00012	nop	00012	nop
00013	nop	00013	nop
00014	ret	00014	ret
00015	mutation_examples.JID.ClassA .ctor(System.Int32 a)	00015	mutation_examples.JID.ClassA .ctor(System.Int32 a)
00019	ldarg.0	00019	ldarg.0
00020	ldc.i4.5	00020	ldc.i4.5
00021	stfld System.Int32 mutation_examples.JID.ClassA::a	00021	pop
00022		00022	pop
00023	ldarg.0	00023	ldarg.0
00024	call System.Void System.Object::.ctor()	00024	call System.Void System.Object::.ctor()
00025	nop	00025	nop
00026	nop	00026	nop
00027	ldarg.0	00027	ldarg.0
00028	ldarg.1	00028	ldarg.1
00029	stfld System.Int32 mutation_examples.JID.ClassA::a	00029	stfld System.Int32 mutation_examples.JID.ClassA::a
00030	nop	00030	nop
00031	ret	00031	ret

IPC operator *(Explicit call of parent's constructor deletion)*

Original C# code:	Mutated C# code:
public class Vehicle	**public class** Vehicle
{ **private** int x;	{ **private** int x;
public Vehicle() {…}	**public** Vehicle() {…}
public Vehicle(**int** x)	**public** Vehicle(**int** x)
{ this.x = x; }	{ this.x = x; }
}	}
public class Car:Vehicle	**public class** Car:Vehicle
{ **public** Car(**int y**)	{ **public** Car(**int y**)
:base(y)	
{…}	{…}
}	}

Pre: Base class defines its non-parametric constructor

IPC operator *(Explicit call of parent's constructor deletion)*

Original CIL code:	Mutated CIL code:

….
instance void .ctor(int32 b)
…
{
 .maxstack 8
  IL_0000: ldarg.0
  IL_0001: ldarg.1
  IL_0002: call instance void Operators.Car::.ctor(int32)
………
}
	….
	instance void .ctor(int32 b)
	…
	{
	.maxstack 8
	  IL_0000: ldarg.0
	  IL_0001: call instance void Operators.Car::.ctor()
	…………..
	}

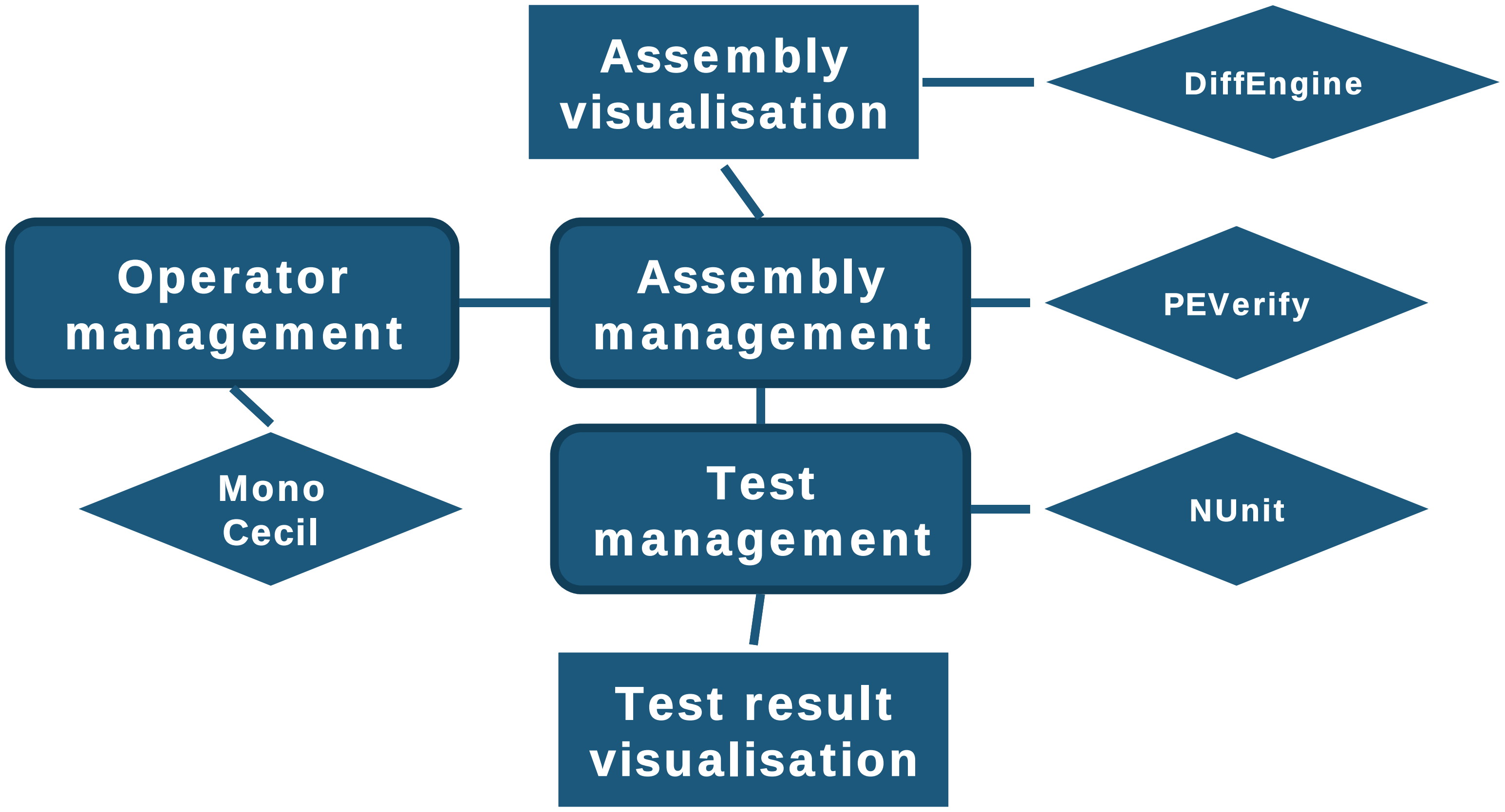
ILMutator system

- **I**ntermediate **L**anguage Mutator supports mutation of programs in .NET environment
- Introduces standard and object-oriented mutations in the intermediate code derived from compiled C# programs using Mono.Cecil library
- User can view the original intermediate code and the mutated code with highlighted differences

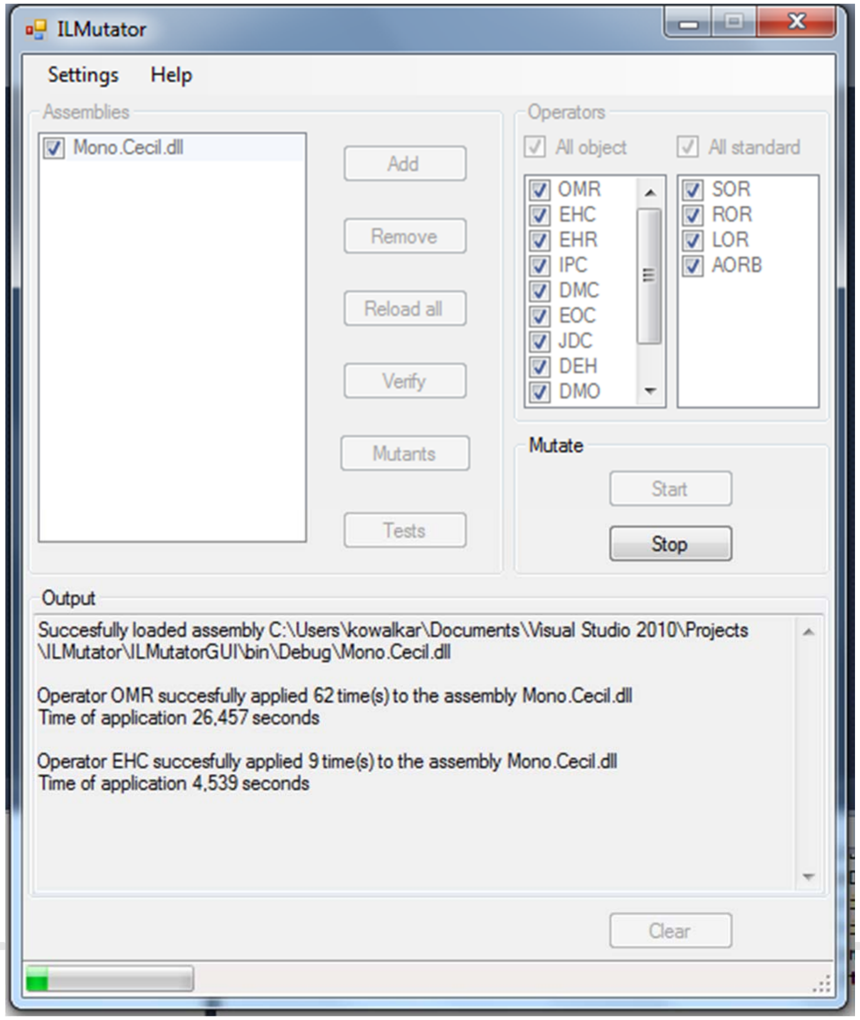
ILMutator system

- Execution of tests on the original and mutated assemblies (NUnit)
- Verification of mutated assemblies with PEVerify tool (delivered with .NET Framework)
- Implements 4 standard and selected object-oriented operators

ILMutator system - architecture

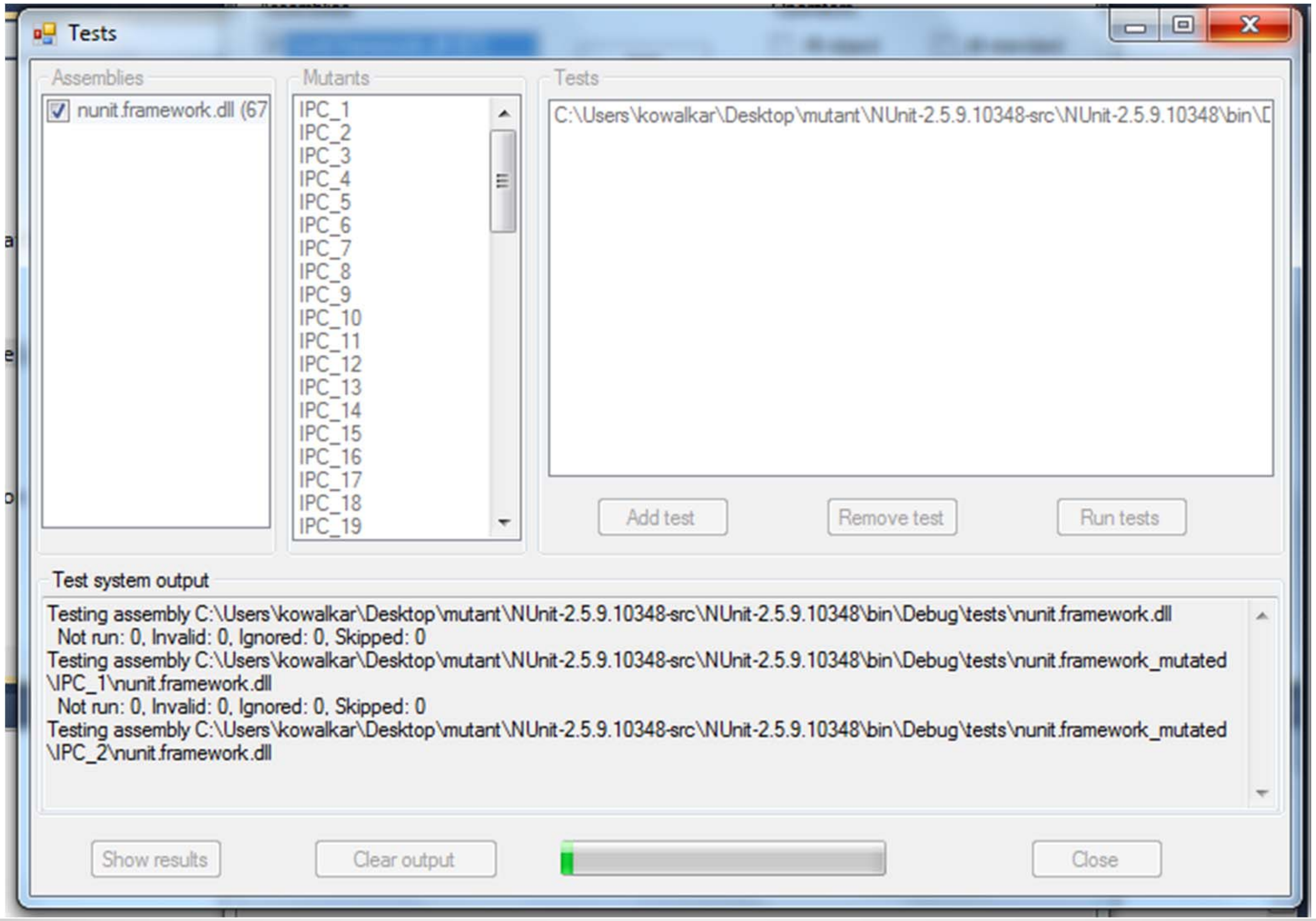


ILMutator system – during work



```
Succesfully loaded assembly C:\Users\kowalkar\Documents\Visual Studio 2010\Projects
\ILMutator\ILMutatorGUI\bin\Debug\Mono.Cecil.dll

Operator OMR succesfully applied 62 time(s) to the assembly Mono.Cecil.dll
Time of application 26,457 seconds

Operator EHC succesfully applied 9 time(s) to the assembly Mono.Cecil.dll
Time of application 4,539 seconds
```

ILMutator system – test runner



```
C:\Users\kowalkar\Desktop\mutant\NUnit-2.5.9.10348-src\NUnit-2.5.9.10348\bin\D
```

```
Testing assembly C:\Users\kowalkar\Desktop\mutant\NUnit-2.5.9.10348-src\NUnit-2.5.9.10348\bin\Debug\tests\nunit.framework.dll
 Not run: 0, Invalid: 0, Ignored: 0, Skipped: 0
Testing assembly C:\Users\kowalkar\Desktop\mutant\NUnit-2.5.9.10348-src\NUnit-2.5.9.10348\bin\Debug\tests\nunit.framework_mutated
\IPC_1\nunit.framework.dll
 Not run: 0, Invalid: 0, Ignored: 0, Skipped: 0
Testing assembly C:\Users\kowalkar\Desktop\mutant\NUnit-2.5.9.10348-src\NUnit-2.5.9.10348\bin\Debug\tests\nunit.framework_mutated
\IPC_2\nunit.framework.dll
```

Experiments – mutation operators

EOC – reference comparison and content comparison replacement

IPC – explicit call of a parent’s constructor deletion

JDC – C# supported default constructor create

JID – member variable initialization deletion

OMR – overloading method contents change

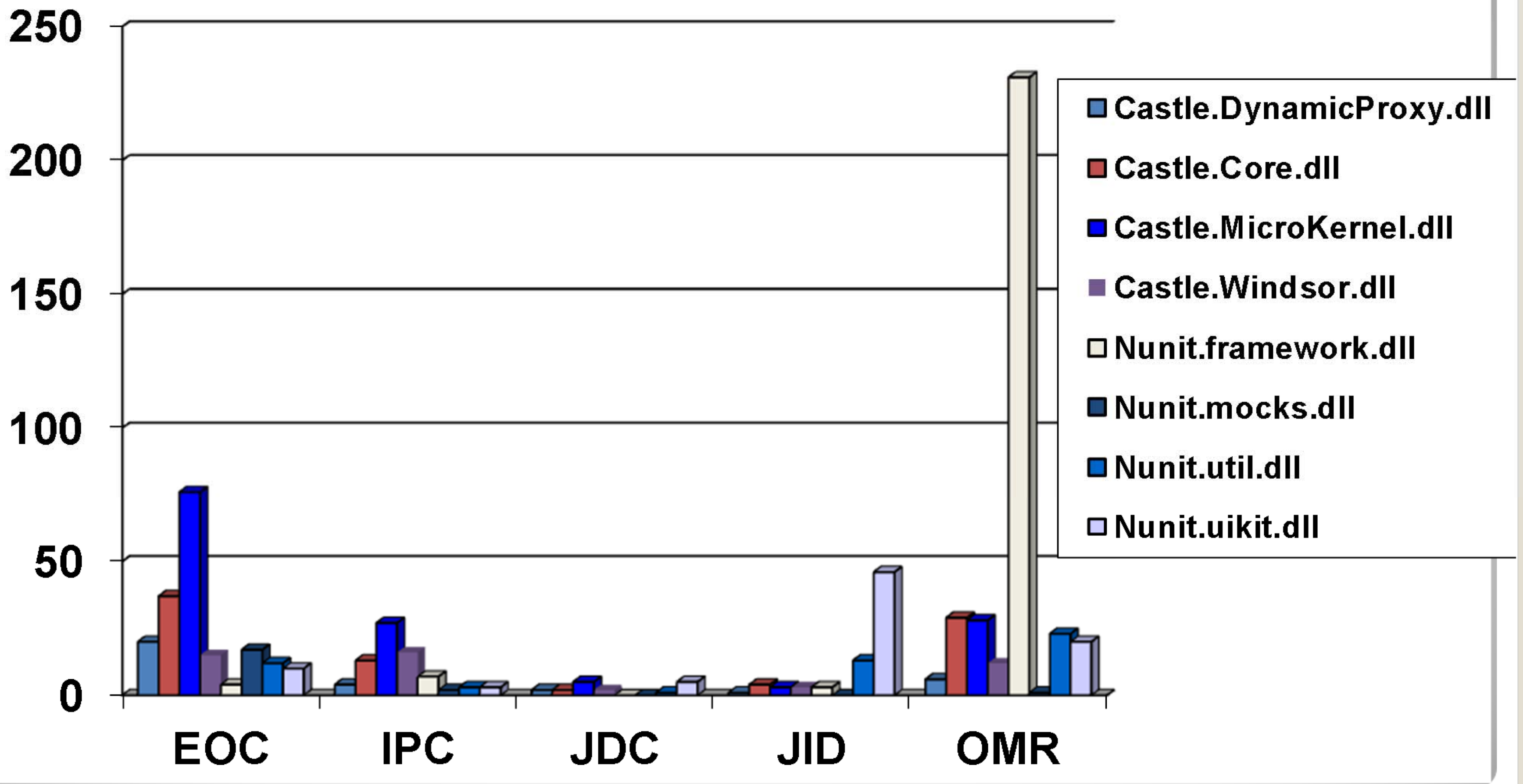
PNC - new method call with child class type

0 mutants

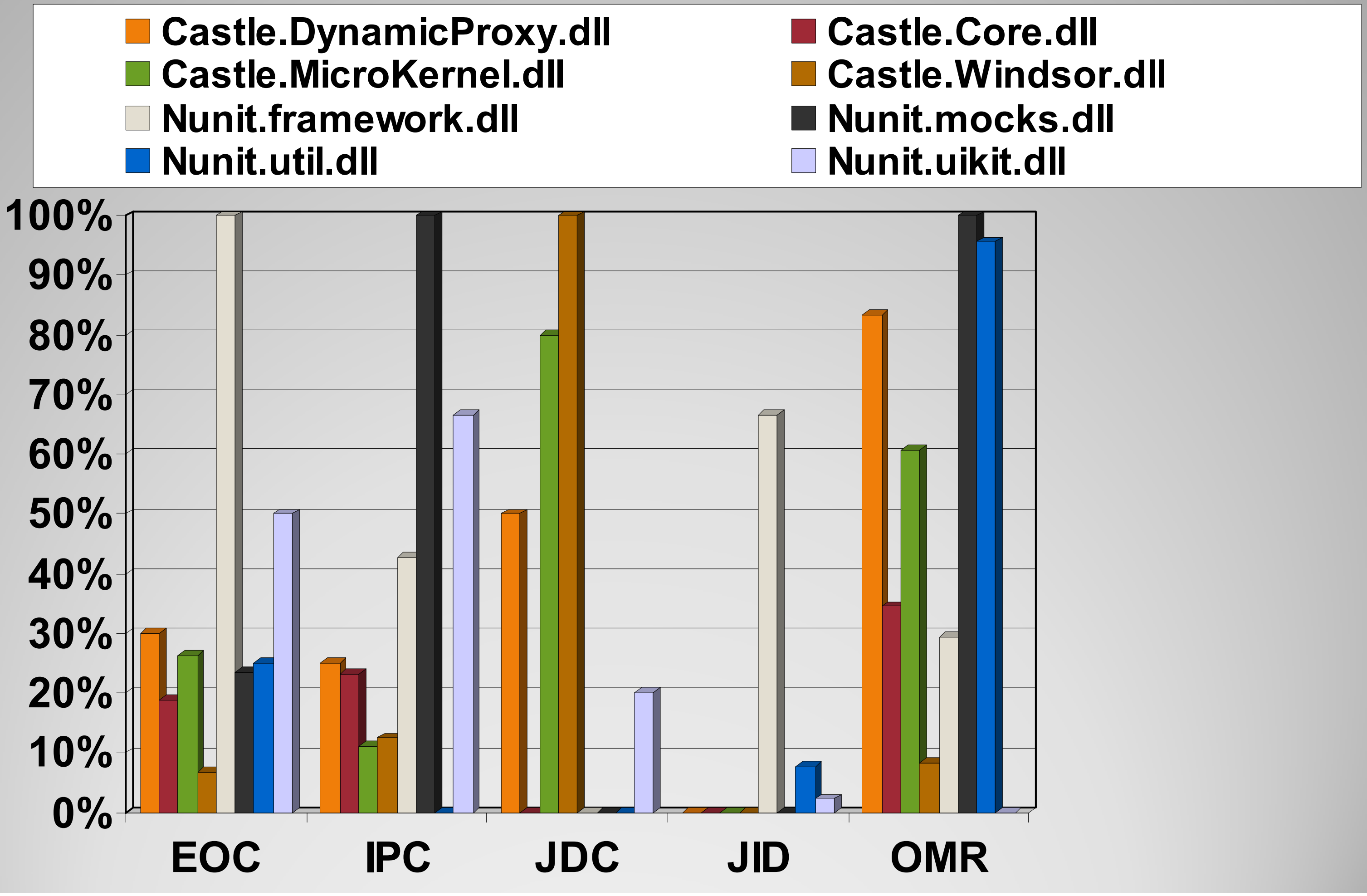
Experiments - mutated assemblies

	Program	Size [kB]	LOC	Classes	Unit tests
1	Castle.Dynamic Proxy	76	5036	71	82
2	Castle.Core	60	6119	50	171
3	Castle.Micro Kernel	112	11007	86	88
4	Castle.Wiondsor	64	4240	34	92
5	Nunit.framework	40	4415	37	397
6	NUnit.mock	20	579	6	42
7	NUnit.util	88	6405	34	211
8	NUnit.uikit	352	7556	30	32

Results – number of mutants



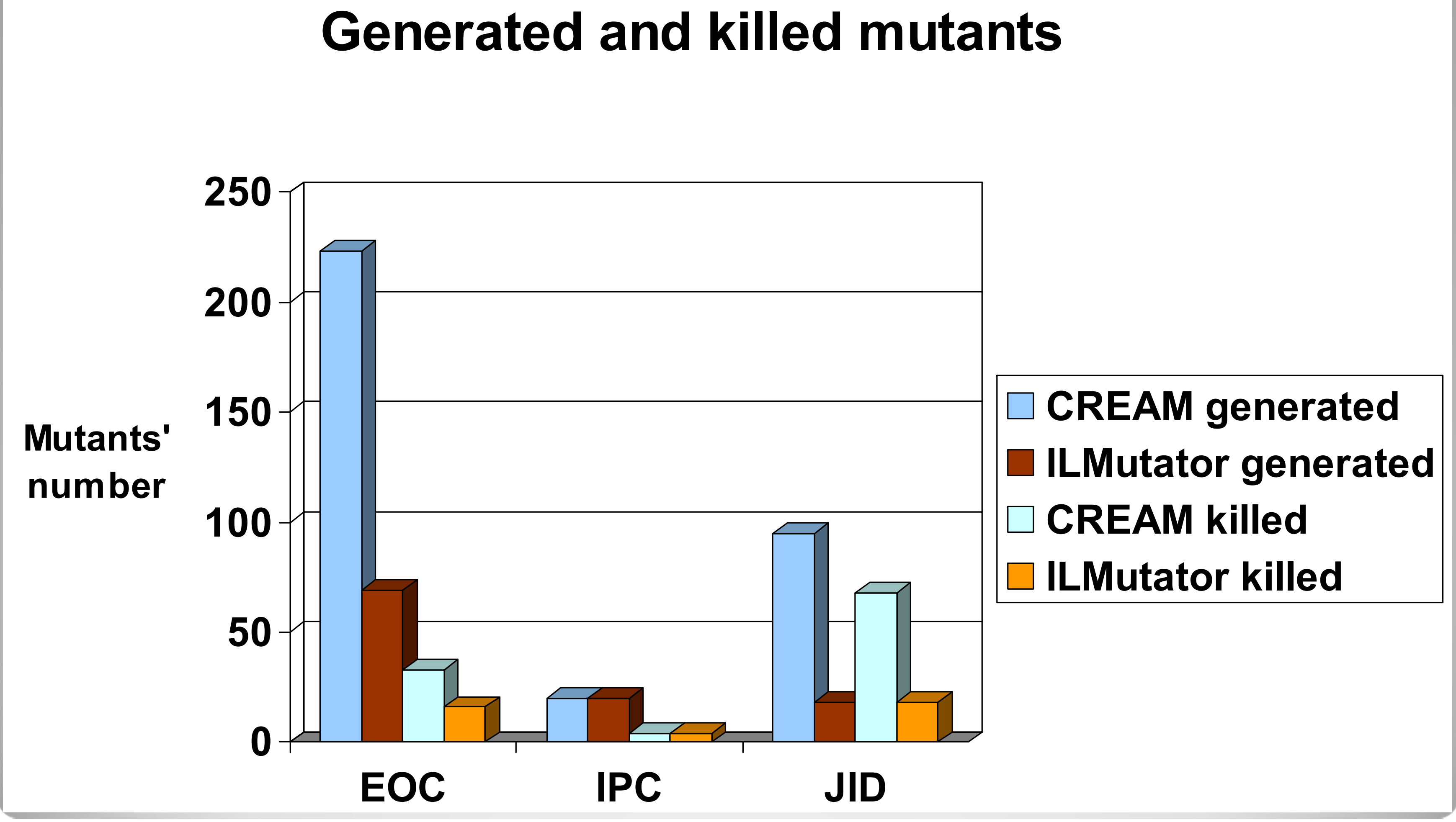
Results – mutation score



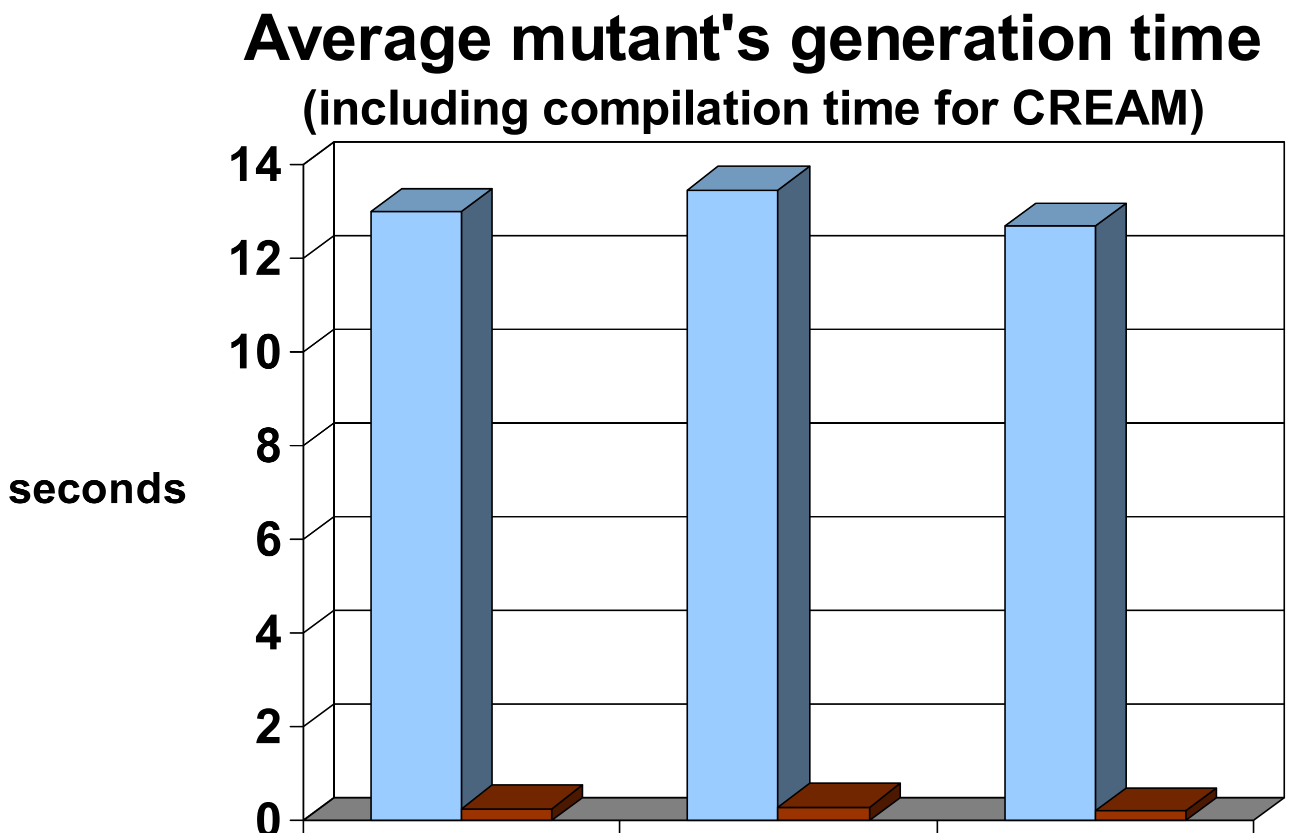
CREAM system

- Parser based CREAtor of Mutants
- Applies standard and object-oriented operators
- Uses compilation and reflection mechanisms
- Tests mutants with unit test frameworks

Comparison with CREAM 2.0



Comparison with CREAM 2.0



	EOC	IPC	JID
🟦 CREAM 2.0	13	13,45	12,68
🟥 ILMutator	0,27	0,3	0,22

Conclusions

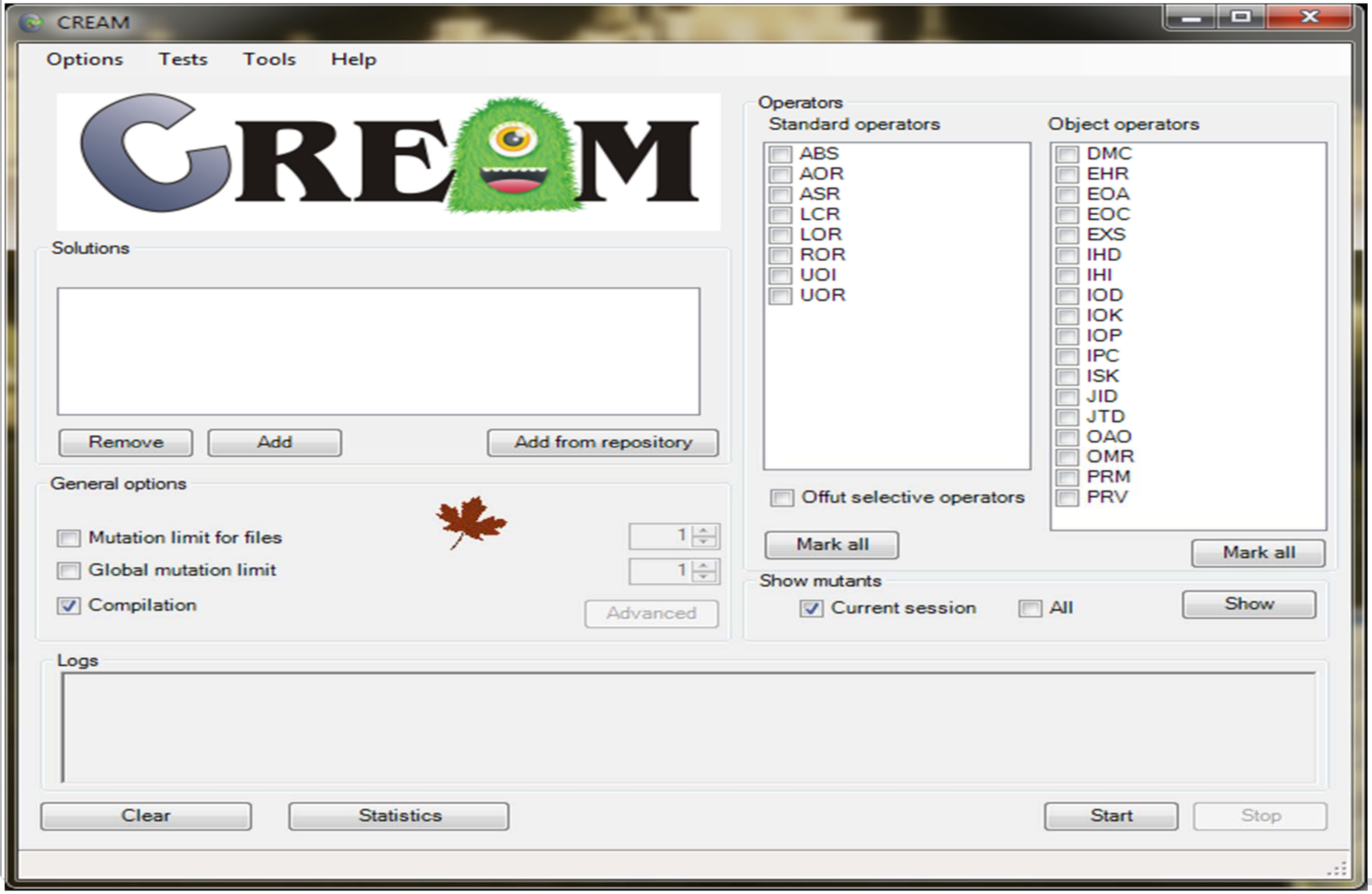
- Introducing mutations on the intermediate language level – more efficient, faster
- Mutated program doesn’t have to be compiled
- Identification of mutation locations - more effort to implement
- Lack of compilation - necessity of correctness checking

Future work

- More mutation operators
- Other ways of generating and storing mutants (e.g. metamutant)
- Other methods of testing (not only unit tests)
- New versions of libraries (Mono.Cecil 0.9) or other libraries (Microsoft.CCI) for mutation injection
- Better visualization of mutated code (CIL<->C#)
- Identification of equivalent mutants

Q&A

CREAM – main window



Operators

Standard operators: [ ] ABS, [ ] AOR, [ ] ASR, [ ] LCR, [ ] LOR, [ ] ROR, [ ] UOI, [ ] UOR

[ ] Offut selective operators

Object operators: [ ] DMC, [ ] EHR, [ ] EOA, [ ] EOC, [ ] EXS, [ ] IHD, [ ] IHI, [ ] IOD, [ ] IOK, [ ] IOP, [ ] IPC, [ ] ISK, [ ] JID, [ ] JTD, [ ] OAO, [ ] OMR, [ ] PRM, [ ] PRV

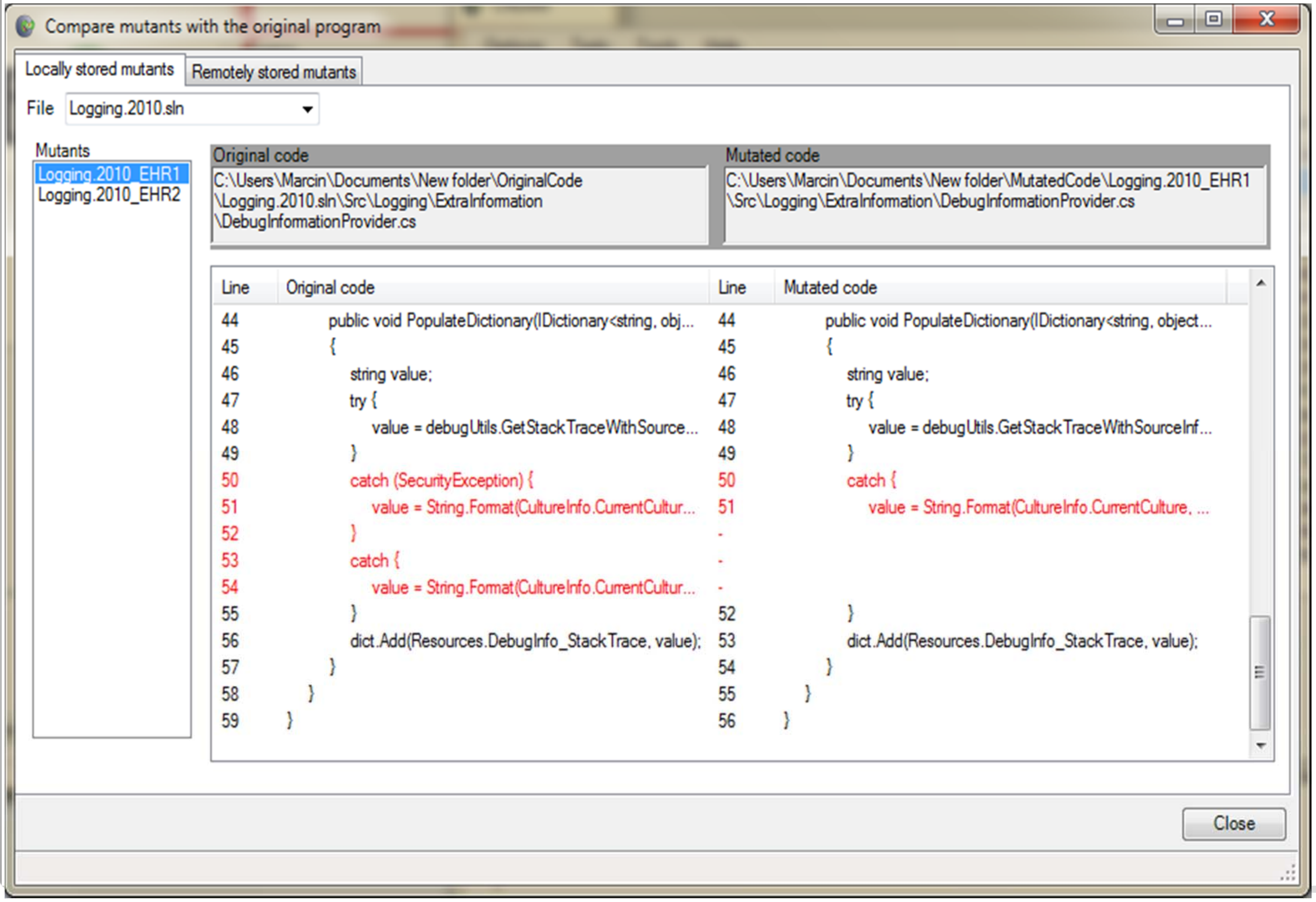
[ ] Mutation limit for files 1

[ ] Global mutation limit 1

[✓] Compilation

Show mutants: [✓] Current session [ ] All

CREAM – original and mutated code



Original code
C:\Users\Marcin\Documents\New folder\OriginalCode\Logging.2010.sln\Src\Logging\ExtraInformation\DebugInformationProvider.cs

Mutated code
C:\Users\Marcin\Documents\New folder\MutatedCode\Logging.2010_EHR1\Src\Logging\ExtraInformation\DebugInformationProvider.cs

```
Line  Original code                                          Line  Mutated code
44        public void PopulateDictionary(IDictionary<string, obj...   44        public void PopulateDictionary(IDictionary<string, object...
45        {                                                   45        {
46            string value;                                   46            string value;
47            try {                                           47            try {
48                value = debugUtils.GetStackTraceWithSource...   48                value = debugUtils.GetStackTraceWithSourceInf...
49            }                                               49            }
50            catch (SecurityException) {                     50            catch {
51                value = String.Format(CultureInfo.CurrentCultur...   51                value = String.Format(CultureInfo.CurrentCulture, ...
52            }                                               -
53            catch {                                         -
54                value = String.Format(CultureInfo.CurrentCultur...   -
55            }                                               52            }
56            dict.Add(Resources.DebugInfo_StackTrace, value);   53            dict.Add(Resources.DebugInfo_StackTrace, value);
57        }                                                   54        }
58    }                                                       55    }
59 }                                                          56 }
```