Power and productivity
for a better world™

A Short Introduction to Randoop.NET

Abstract: This document contains an introduction to using Randoop.NET: a tool for automatically generating regression test suites for C# programs. It contains simple examples that demonstrate how to install and use the tool from within the Visual Studio IDE.

Xiao Qu, Ph.D.

ABB Corporate Research

6/4/2015

Introduction

A common use of *unit test suites* is to guard against *regression defects*. A regression test suite consists of a number of tests that (mostly) pass on the current version of the program. This suite is then run on future versions to ensure that the behavior of the program has not changed unless it is designed to change.

Randoop.NET is a tool for automatically generating *regression test suites* for existing codebases, it works on C# code (a version of Randoop which works on Java code is at http://mernst.github.io/randoop/), built as a visual studio add-in. Original Randoop.NET is a command line based tool, available from CodePlex (http://randoop.codeplex.com/). But this version did not implement advanced features such as regression assertions, nor was it able to merge single test cases into a test suite in *MSTest* format, while one advantage of the add-in is to merge all single test cases into one test project in MSTest format, thus one can run the tests directly from the VS IDE, with the ability to collect test information such as the code coverage.

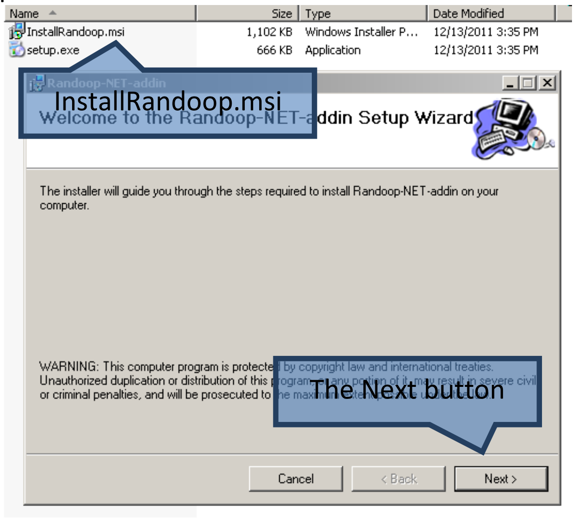
Prerequisites

Randoop.NET requires the VS2008 (compatible with VS2010) and .Net framework v3.5 or later. Randoop.NET works on .NET assemblies (C# programs); other languages are not supported.

Installing Randoop.NET

The installer of a Visual Studio add-in (with the Randoop.NET wrapped in) is available after you build the project (i.e., randoop-VS-plugin-2010\Randoop\InstallRandoop\). To install the visual studio add-in, just run the installer "InstallRandoop.msi" in the release directory. See Figure 1 for details.

1. Double click **InstallRandoop.msi** and then press the **"Next>"** button



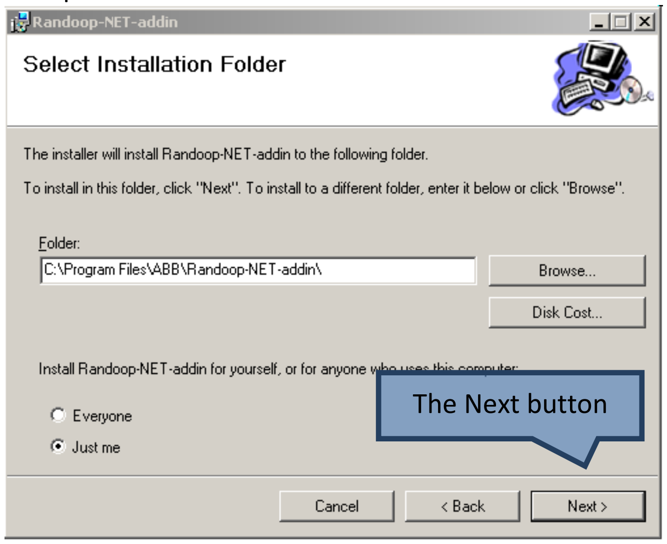
Name	Size	Type	Date Modified
InstallRandoop.msi	1,102 KB	Windows Installer P...	12/13/2011 3:35 PM
setup.exe	666 KB	Application	12/13/2011 3:35 PM

Welcome to the Randoop-NET-addin Setup Wizard

The installer will guide you through the steps required to install Randoop-NET-addin on your computer.

WARNING: This computer program is protected by copyright law and international treaties. Unauthorized duplication or distribution of this program, or any portion of it, may result in severe civil or criminal penalties, and will be prosecuted to the maximum extent possible under the law.

2. Select the folder in which the add-in is installed and then press the **"Next>"** button



Select Installation Folder

The installer will install Randoop-NET-addin to the following folder.

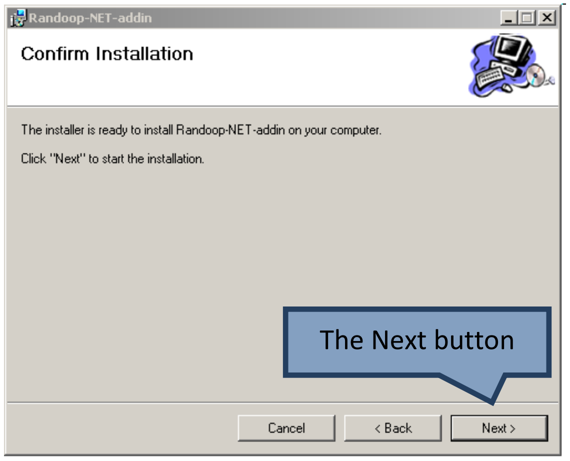
To install in this folder, click "Next". To install to a different folder, enter it below or click "Browse".

Folder: C:\Program Files\ABB\Randoop-NET-addin\

Install Randoop-NET-addin for yourself, or for anyone who uses this computer:

( ) Everyone
[✓ Just me]

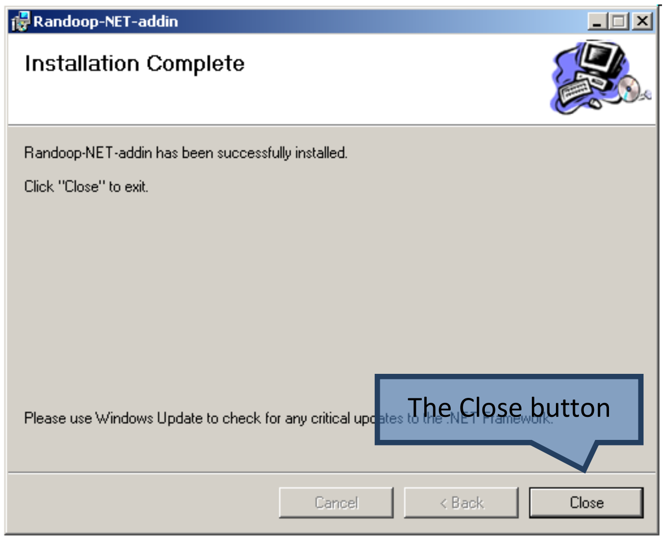
3. Confirm installation and then press **Next**



The installer is ready to install Randoop-NET-addin on your computer.

Click "Next" to start the installation.

4. Installation complete and then press **Close**



Randoop-NET-addin has been successfully installed.

Click "Close" to exit.

Please use Windows Update to check for any critical updates to the .NET Framework.

Figure 1. Installing the Randoop.NET VS add-in

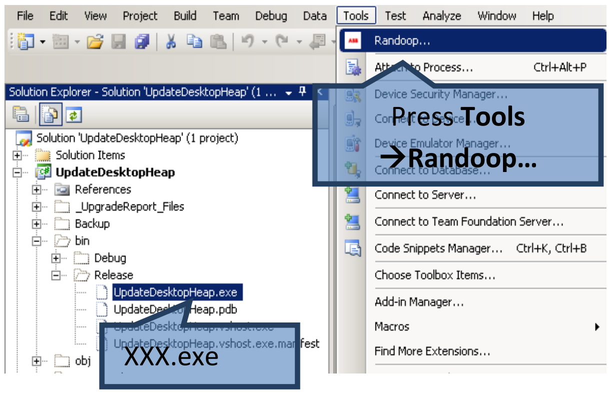
Generating tests with Randoop.NET

With the Randoop.NET Visual Studio add-in installed, you can select a .NET assembly in your project and generate tests (in *MSTest* format) from it. You then run the tests by selecting the tests you are interested in, right clicking, and selecting "**Run Tests"** provided by Visual Studio. All steps are illustrated in Figure 2.

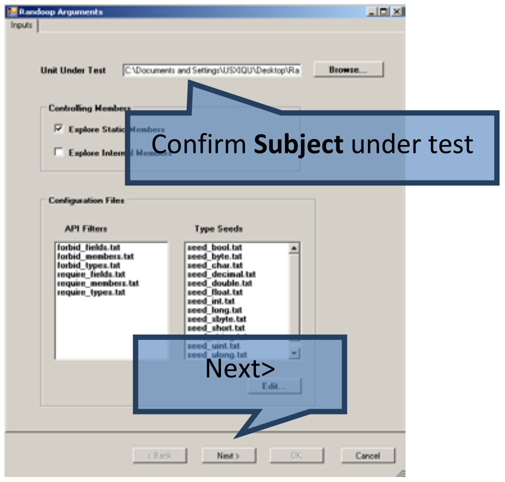
In order to optimize the test generation process of Randoop.NET, so as to improve the cost-effectiveness of the test suites generated by Randoop.NET, we also provide some auxiliary tools that work before (pre-) or after (post-) generation. They are introduced in section of **Auxiliary Tools**.

Randoop.NET tests code by executing random invocations of API calls. This can lead to potentially undesirable behaviors—if the code under test can delete files, open connections, send items to a printer, etc. then there is a chance that Randoop.NET will invoke the code that performs these actions with arbitrary input parameters. We strongly recommend running Randoop.NET under a controlled environment (e.g. machines reserved for testing), not under normal development machines. The examples in this manual, however, are safe to run on any machine.

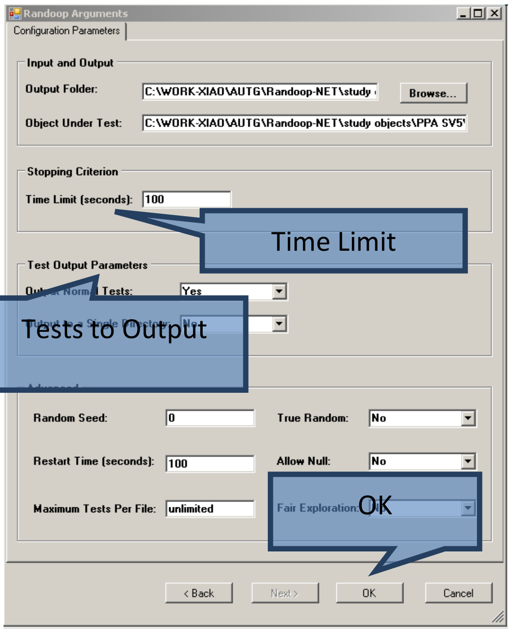
1. Select .NET assembly (*.exe, *.dll) and then press **Tools → Randoop…**


File Edit View Project Build Team Debug Data Tools Test Analyze Window Help
Randoop…
Attach to Process… Ctrl+Alt+P
Device Security Manager…
Device Emulator Manager…
Connect to Database…
Connect to Server…
Connect to Team Foundation Server…
Code Snippets Manager… Ctrl+K, Ctrl+B
Choose Toolbox Items…
Add-in Manager…
Macros
Find More Extensions…
Solution Explorer - Solution 'UpdateDesktopHeap' (1 …
Solution 'UpdateDesktopHeap' (1 project)
Solution Items
UpdateDesktopHeap
References
_UpgradeReport_Files
Backup
bin
Debug
Release
UpdateDesktopHeap.exe
UpdateDesktopHeap.pdb
UpdateDesktopHeap.vshost.exe
UpdateDesktopHeap.vshost.exe.manifest
obj
Press Tools
→Randoop…
XXX.exe

2. Confirm Object under test, edit configuration files if necessary [A], and then press **Next>**



3. Customize basic Randoop arguments/options [B] such as what tests to output and time to stop, and then press **OK**



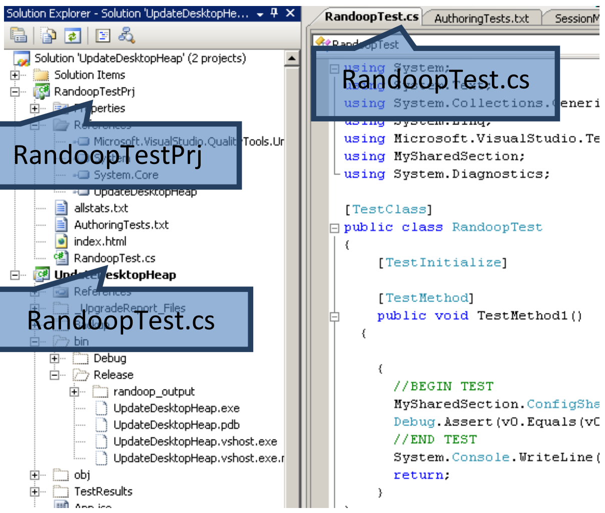
Input and Output
Output Folder: C:\WORK-XIAO\AUTG\Randoop-NET\study
Object Under Test: C:\WORK-XIAO\AUTG\Randoop-NET\study objects\PPA SV5

Stopping Criterion
Time Limit (seconds): 100

Test Output Parameters
Output Normal Tests: Yes
Output to a Single Directory: No

Advanced
Random Seed: 0
True Random: No
Restart Time (seconds): 100
Allow Null: No
Maximum Tests Per File: unlimited
Fair Exploration: No

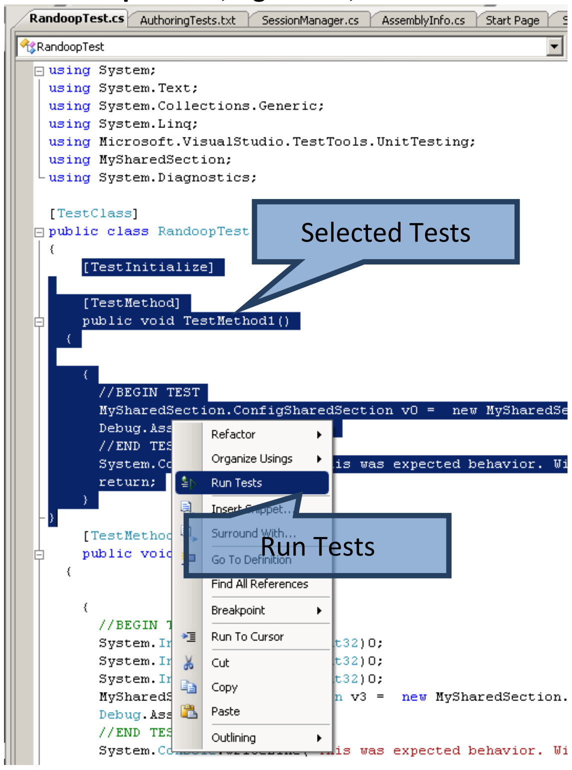
4. When you select **OK**, the Randoop.NET will begin generating tests. The tests will be placed in file **RandoopTest.cs**, which is included in a test project **RandoopTestPrj** under the same solution of the object under test.



```
using System;
using System.Text;
using System.Collections.Generi
using System.Linq;
using Microsoft.VisualStudio.Te
using MySharedSection;
using System.Diagnostics;

[TestClass]
public class RandoopTest
{
    [TestInitialize]

    [TestMethod]
    public void TestMethod1()
  {

    {
      //BEGIN TEST
      MySharedSection.ConfigSha
      Debug.Assert(v0.Equals(vC
      //END TEST
      System.Console.WriteLine(
      return;
    }
```

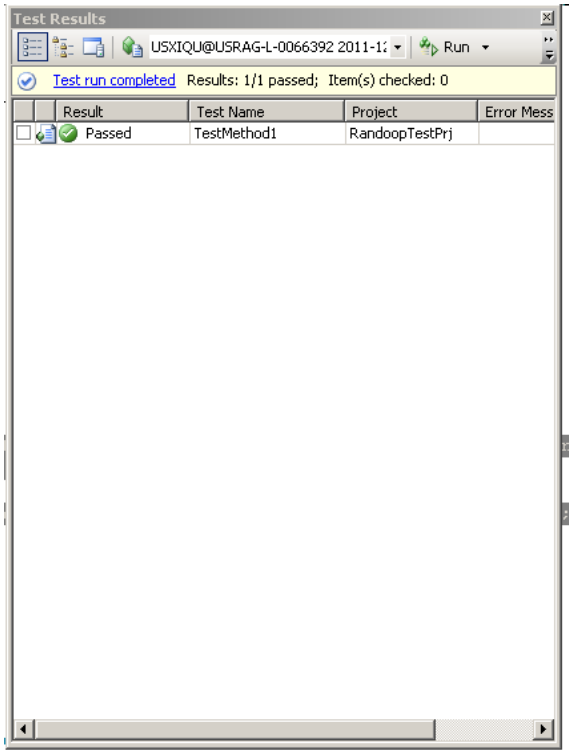
A, B: default configuration files and options are applicable in most cases, but advanced users may want to customize these options to generate more "intelligent" test cases (**for example, specifying which classes/methods are to be tested**). All these options are described in details in Section **Advanced Usage – Customizing Options**

5. Select test cases that you are interested in the **RandoopTest.cs**, right click, and select **Run Tests**



```
using System;
using System.Text;
using System.Collections.Generic;
using System.Linq;
using Microsoft.VisualStudio.TestTools.UnitTesting;
using MySharedSection;
using System.Diagnostics;

[TestClass]
public class RandoopTest
{
    [TestInitialize]

    [TestMethod]
    public void TestMethod1()
  {

      {
        //BEGIN TEST
        MySharedSection.ConfigSharedSection v0 =  new MySharedSe
        Debug.Ass
        //END TES
        System.Co                           is was expected behavior. Wi
        return;
      }
}
    [TestMethoc
    public voic
  {

      {
        //BEGIN T
        System.Ir                           t32)0;
        System.Ir                           t32)0;
        System.Ir                           t32)0;
        MySharedS                           n v3 =  new MySharedSection.
        Debug.Ass
        //END TES
        System.C                            is was expected behavior. Wi
```

6. Run Tests and then check results



Test run completed Results: 1/1 passed; Item(s) checked: 0

Result	Test Name	Project	Error Mess
Passed	TestMethod1	RandoopTestPrj	

Figure 2. Generating and Running Tests with Randoop.NET

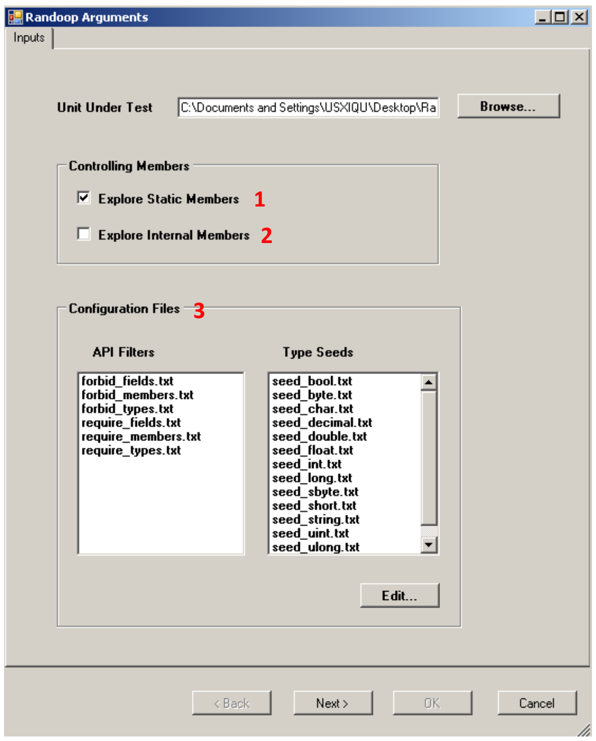
Advanced Usage – Customizing Options

(Most contents are copied from **Randoop Manual.doc** available at http://randoop.codeplex.com/documentation)

By default, Randoop.NET explores the *public* members declared in the given assemblies by creating sequences of API calls and executing them.

Randoop.NET has several options to control its behavior. The options are classified into the following categories.

- Controlling members that are allowed/forbidden (in configuration files).
- Controlling test generation.
- Controlling which and where tests are output.



Randoop Arguments

Inputs

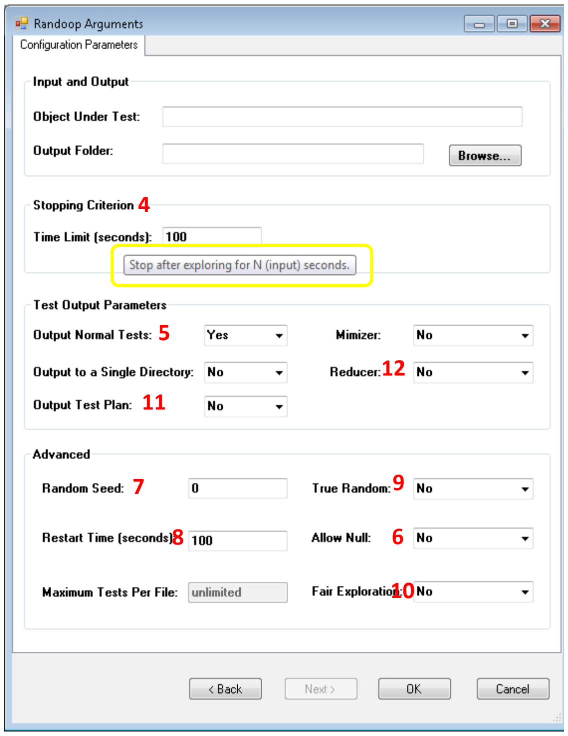
Unit Under Test: C:\Documents and Settings\USXIQU\Desktop\Ra

Controlling Members

- [✓] Explore Static Members 1
- [ ] Explore Internal Members 2

Configuration Files 3

API Filters	Type Seeds
forbid_fields.txt	seed_bool.txt
forbid_members.txt	seed_byte.txt
forbid_types.txt	seed_char.txt
require_fields.txt	seed_decimal.txt
require_members.txt	seed_double.txt
require_types.txt	seed_float.txt
	seed_int.txt
	seed_long.txt
	seed_sbyte.txt
	seed_short.txt
	seed_string.txt
	seed_uint.txt
	seed_ulong.txt



Randoop Arguments

Configuration Parameters

Input and Output

Object Under Test:

Output Folder:

Stopping Criterion 4

Time Limit (seconds): 100

Stop after exploring for N (input) seconds.

Test Output Parameters

Output Normal Tests: 5	Yes	Mimizer:	No
Output to a Single Directory:	No	Reducer: 12	No
Output Test Plan: 11	No		

Advanced

Random Seed: 7	0	True Random: 9	No
Restart Time (seconds) 8	100	Allow Null: 6	No
Maximum Tests Per File:	unlimited	Fair Exploration: 10	No

Figure 3. Randoop.NET Options

The following options control the API entities (types, constructors, methods and fields) that Randoop.NET uses to create tests (a short description of each option will be shown when the mouse pauses on each option's "hover rectangle", as highlighted in the right window of Figure 3).

1. **Explore Static Members**: default is to explore both static and instance members; otherwise "don't explore static members".
2. **Explore Internal Members**: Explore internal members in addition to public members. Default is to explore only public members. Note that if this option is given, the generated tests may not compile.
3. **Configuration Files:** There are 20 possible configuration files (details can be found at Section Configuration Files). Lines starting with "#" are comments, ignored in configuration files.
   - **API Filters:** For these 6 files, each line in the file is interpreted as a wildcard pattern. The pattern should not contain whitespace (if it does, it will be removed before applying the pattern). A wildcard pattern can contain '*' characters. A '*' matches any sequence of strings, including the empty string.
   - **Type Seeds:** The remaining 14 files specify simple type values (or strings) to use when calling an API that requires a simple type as input. The values for a given type are specified in a file, and each line in the file should contain a string representation of the value that can be parsed into the given type.
4. **Time Limit:** Stop after exploring for N seconds. Default: 100 seconds.
5. **Output Normal Test:** Output test inputs that lead to non-exceptional behavior. Default: Yes.
6. **Allow Null:** Allow the use of null as a parameter. Default behavior never uses null as a parameter.
7. **Random Seed:** Use N as the initial random seed. Round 1 of generation uses N as seed, round 2 uses N+1, etc. Default: 0.
8. **Restart Time:** Kill the generation process and spawn a new one every N seconds. Default is N=100 seconds.

9. **True Random:** Use random seed from *System.Security.Cryptography.RandomNumberGenerator*. Note: if you use this option, the “random seed” option has no effect. Default is to use *System.Random*.
10. **Fair Exploration:** Uses a fair algorithm for exploring the object space. Turned off by default.
11. **Output Test Plan:** Beside test case code, the tool also outputs what classes/methods are tested by each test case.
12. **Test Reducer:** Reduce test redundancy. More details are described in Test Reducer

Configuration Files

Randoop.NET uses a set of configuration files to determine which types, methods, etc. to generate tests for, and which primitive values to use as inputs to methods. There are 20 configuration files.

API Filters. For the next 6 files, each line in the file is interpreted as a wildcard pattern. The pattern should not contain whitespace (if it does, it will be removed before applying the pattern). A wildcard pattern can contain '*' characters. A '*' matches any sequence of strings, including the empty string.

Examples of wildcard patterns:

`System.Xml*` matches any string that starts with "System.Xml"

`*Reflection*` matches any string that contains "Reflection"

For more examples see “ <Installation root>\Randoop-NET-addin\Randoop-NET-release\default_config_files”.

1. **require_types.txt.** Use only types that match at least one wildcard pattern in the given file. The pattern is matched against the result of calling T.ToString() on each type.
2. **require_members.txt.** Use only methods and constructors that match at least one wildcard pattern in the given file. The pattern is matched against the result of calling M.ToString()/returntype on each member.
3. **require_fields.txt.** Use only fields that match at least one wildcard pattern in the given file. The pattern is matched against the result of calling F.ToString() on each field.
4. **forbid_types.txt.** Avoid types that match at least one wildcard pattern in the given file. The pattern is matched against the result of calling T.ToString() on each type.
5. **forbid_members.txt.** Avoid methods and constructors that match at least one wildcard pattern in the given file. The pattern is matched against the result of calling M.ToString() on each member.

6. **forbid_fields.txt.** Avoid fields that match at least one wildcard pattern in the given file. The pattern is matched against the result of calling F.ToString() on each field.

Determining the set of entities under test. *Below is a more precise description of the steps used to determine the entities used to create tests.*

Steps i-iv determine the types used to create tests.

- *i.* *Collect all the public types in the assemblies given as arguments to Randoop. Call the resulting set of types TS.*
- *ii.* *If /internal is given, add all internal types to TS.*
- *iii.* *Remove from TS any type that does not match a wildcard specified in the require_types.txt file.*
- *iv.* *Remove from TS any type that matches a wildcard specified in the forbid_types.txt file.*

Step v determines the methods, constructors and fields used to create tests.

- *v.* *For each type in TS:*
  - *a.* *Collect all the public methods, constructors and fields declared by the types in TS. Call the resulting set MS.*
  - *b.* *If /nostatic is given, remove all static members from MS.*
  - *c.* *Remove any method or constructor that does not match a wildcard specified in the require_members.txt file. Similarly for require_fields.txt*
  - *d.* *Remove any method or constructor that matches a wildcard specified in the forbid_membres.txt file. Similarly for forbid_fields.txt.*

Simple type seeds. The remaining 14 files specify simple type values (or strings) to use when calling an API that requires a simple type as input. The values for a given type are specified in a file, and each line in the file should contain a string representation of the value that can be parsed into the given type.

For example, the file **seed_int.txt** must contain one ***int*** per line, such as "3" or "-1". More specifically, for simple type *T*, each line should contain a string that can be passed to *T.Parse(string)*. The exception is the file **seed_string.txt**, where each line can contain any string.

7. **seed_sbyte.txt**
8. **seed_byte.txt**
9. **seed_short.txt**
10. **seed_ushort.txt**
11. **seed_int.txt**
12. **seed_uint.txt**
13. **seed_long.txt**
14. **seed_ulong.txt**
15. **seed_char.txt**
16. **seed_float.txt**
17. **seed_double.txt**
18. **seed_bool.txt**
19. **seed_decimal.txt**
20. **seed_string.txt**

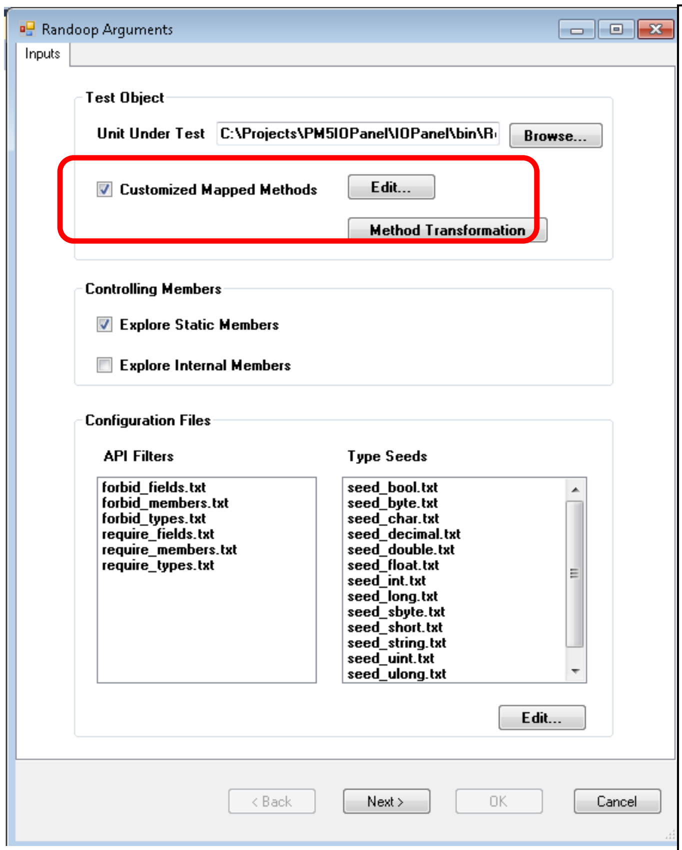
Auxiliary Tools

Pre-generation

Method Transformer

Method transformer lets users delete pre-defined method calls built in the assembly or replace them with pre-defined alternative method calls.

This functionality is used when some particular method calls in the assembly may prevent the automation of test generation. For example, if the methods explored by Randoop invoke a pop-up window (i.e., a messagebox) by calling *MessageBox.Show()*, the generation process may wait for manual intervention (such as close or cancel the messagebox) forever (until the time is out) in the background. This problem prevents Randoop from further exploring other methods in the project. As a result, a big proportion of methods cannot be reached by Randoop, which will be not covered by the generated test suites. To overcome this problem, the user can either specify to delete all *MessageBox.Show* calls in the assembly or to transform the *MessageBox.Show* to *Console.Writeline*, which will redirect the message to console without waiting for any manual intervention.



Unit Under Test: C:\Projects\PM5IOPanel\IOPanel\bin\R

[✓] Customized Mapped Methods

[✓] Explore Static Members

[ ] Explore Internal Members

API Filters:
forbid_fields.txt
forbid_members.txt
forbid_types.txt
require_fields.txt
require_members.txt
require_types.txt

Type Seeds:
seed_bool.txt
seed_byte.txt
seed_char.txt
seed_decimal.txt
seed_double.txt
seed_float.txt
seed_int.txt
seed_long.txt
seed_sbyte.txt
seed_short.txt
seed_string.txt
seed_uint.txt
seed_ulong.txt

1. Check the box of "Customized Mapped Methods".
2. Click on "Edit…", a text file named "mapped_methods.txt" will pop up for users' edits (required format is explained in the file with an example).
3. After editing the file (i.e., specifying the method calls to be deleted or replaced), close the file.
4. Click on "Method Transformation", the modification to the assembly will be completed.

NOTES: After transformation,

1. original assembly will be renamed as "x_orig.exe" or "x_orig.dll". ".pdb" file will be changed accordingly if it exists.
2. remember to reassign strong name to the modified assembly if the original assembly uses strong name assignment
3. currently, modified .dll/.exe does not support coverage collection (an open issue on https://github.com/jbevain/cecil/issues/107)

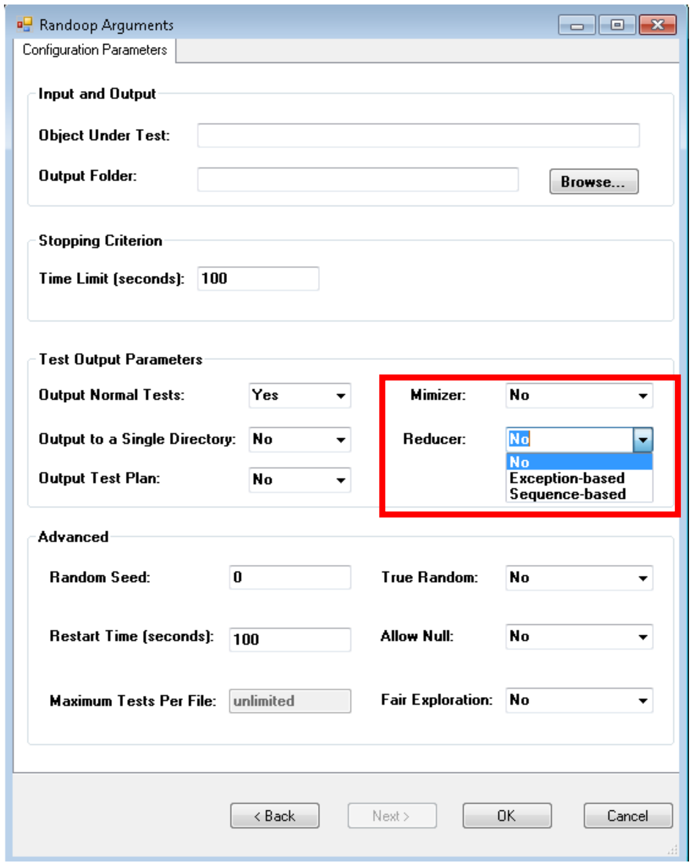
Post-generation

Test Reducer

Test reducer help users reduce the number of generated test cases by moving ***redundant*** test cases. The redundancy is defined in two different ways in this tool:

(1) **exception-based**: two tests are redundant if they end with the same method call *and* throw the same exception. The reducer keeps only one non-redundant test for each method/exception pair;

(2) **sequence-based**: a test case *t* is redundant for a test suite *S* if and only if there exists a test case *t'* from *S*, which is *seq(t)* is a subset of *seq(t')*.



Randoop Arguments

Configuration Parameters

Input and Output

Object Under Test:

Output Folder:

Stopping Criterion

Time Limit (seconds): 100

Test Output Parameters

Output Normal Tests: Yes

Output to a Single Directory: No

Output Test Plan: No

Mimizer: No

Reducer: No

No

Exception-based

Sequence-based

Advanced

Random Seed: 0

Restart Time (seconds): 100

Maximum Tests Per File: unlimited

True Random: No

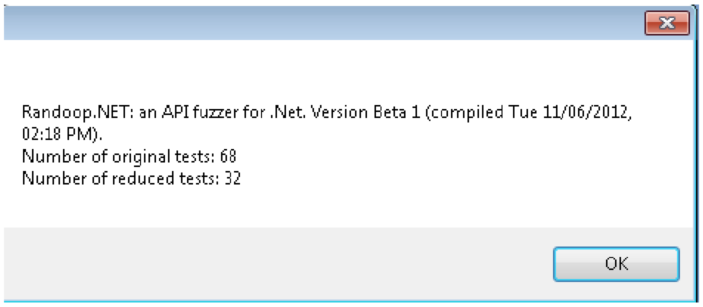
Allow Null: No

Fair Exploration: No

The test reducer is implemented as one of the Randoop generation option as highlighted in the left figure.

The users only need to select one option from the drop-down list before clicking on the "OK" button. The default choice is "no" reduction.

A MessageBox will pop up during the generation telling the number of original test cases and the number of reduced test cases (shown as in the Figure below)



Randoop.NET: an API fuzzer for .Net. Version Beta 1 (compiled Tue 11/06/2012, 02:18 PM).
Number of original tests: 68
Number of reduced tests: 32