|  |
| --- |
| Programmer’s Guide (Pit File Mode) |
| 2013-2014 |
| By Rakotyansky Maria |

Contents

[1. Introduction 3](#_Toc397283807)

[2. Requirements List 3](#_Toc397283808)

[3. Use Case Diagrams and Activity Diagrams 4](#_Toc397283809)

[3.1. Business Level Use Case Diagram 4](#_Toc397283810)

[3.2. System Level Use Case Diagram 6](#_Toc397283811)

[3.3. System Level Activity Diagram 10](#_Toc397283812)

[4. Class Diagram 12](#_Toc397283813)

# Introduction

Evolutionary Fuzzing Framework is implemented in C# in Visual Studio 2010 environment for Window OS. The application can work in 2 modes. The current Guide describes implementation of the Pit File Mode. In the Pit File Mode population is a group of mutated Peach Pit files, produced from the same ancestor. The number of citizens is predefined by the User. The application is quite flexible and all the parameters required by the system can be selected or updated using simple GUI. The resultant data is printed to the output window form. In case of necessity it can also be saved in a separate directory.

# Requirements List

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ID** | **Description** | **Source** | **Functional** | **Non-Functional** |
| 1. | Create simple automatic system for fuzzing target software based on the usage of evolutionary algorithms | Client Story |  | C |
| 2. | Use the Peach Fuzzing Platform as the basics of a new system. | Client Story |  | C |
| 3. | The system must support 2 modes: Pit File Mode and Modified Input Mode | Client Story | O |  |
| 4. | GUI provides mode selection | Client Story | I |  |
| 5. | User inserts paths to save logs as parameters | Client Story | I |  |
| 6. | The system knows to build the command line for Peach Fuzzer on basis of the entered parameters: Test Range and path to Peach Pit file | Client Story | I + O |  |
| 7. | The system knows to create session configuration (Genetic Algorithms) on the basis of entered parameters. | Client Story | I+O |  |
| 8. | Parameters for session configuration: Population Number, Parents Selection Strategy (Naïve, FPS, Tournament), Mutation Rate (0 to 1), Number of Iterations, Survival Strategy (Elitism, Aging, Genitor). | Client Story | D |  |
| 9. | The system is able to verify entered parameters | Client Story | O |  |
| 10. | On entering parameters the user should be able to run the session | Client Story | I+O |  |
| 11. | The fitness function should be based on the following data: test time, PeachPit validity, number of faults | Client Story | O |  |
| 12. | At the end of the session statistics should be written to the xml file in the following format:  <Iteration Number >  <Convergence/>  <Average Fitness/>  <Fault Pits/>  </Iteration Number> | Client Story | O+D |  |
| 13. | Client should be able to see results of the session | Client Story | I+O |  |
| 14. | Results of the session can be returned in the form of chart(according to chosen parameter) | Client Story | I+O |  |
| 15. | Results of the session can be saved in the output files:  Session parameters (pdf format), charts (jpg format), statistics(xml format), peach output files in case of faults (txt format) | Client Story | O+D |  |
| 16. | The GUI should be simple and easy understandable | Client Story | I |  |

# Use Case Diagrams and Activity Diagrams

# Business Level Use Case Diagram

The Business Use-Cases for the current system are as follows:

* The User runs the Evolutionary Fuzzing Framework (Pit File Mode) on target software
* The User views results in the output window or in the form of the text files.

The diagram will look quite simple. See **Figure 3**. It shows the principal actor – the Customer and 2 Use-Cases. See **Table 2**. On this level we do not show inner actors as they are the part of Use-Case. Here is described “black-box”, not how Business Use-Cases are implemented.

**Figure 3:** Business Level Use Case Diagram

Business: “Checking for vulnerabilities”

Customer

**Table 1: Stakeholders and interests on the business level**

|  |  |  |
| --- | --- | --- |
| **Stakeholder** | **Interests** | **Actor?** |
| User | Check target software. View results. | Yes |

**Table 2: Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **UC Code** | **UC Name** | **Actor** | **Description** |
| BUC-1 | Check target software. | User | Run application on the target and start fuzzing process. |
| BUC-2 | View results | User | View results in the special window or in the form of the output files. |

# System Level Use Case Diagram

The current part will show how the user can make use of the system to get the result he wants.

The System Use Case diagram can be seen on **Figure 4**. The description to the diagram is provided in **Table 3**.

**Figure 4:** System Level Use Case Diagram

Peach

­­

User

<<Includes>>

<<Extends>>

<<Extends>>

**Table 3: Description**

|  |  |  |  |
| --- | --- | --- | --- |
| **UC Code** | **UC Name** | **Actor** | **Description** |
| SUC-1 | Enter parameters | User | Providing parameters necessary for running the session. |
| SUC-2 | Run session | User | After choosing parameters session can be started. The session includes the process of fuzzing. The current system is unable to do it alone, it communicates with Peach Fuzzer Platform, which is shown as a separate supporting actor.  Session also includes Validating parameters as subtask |
| SUC -3 | Return output | User | Return results of the current session to the user. As can be seen from the diagram the extending classes Return output as chart and Save output as files add their behavior.  User can see results in the window or in case of necessity save them as files. |

Further is given a text detailed description of every Use-Case.

* **Name**: Enter Parameters System Use case (SUC-1)
* **Brief Description**: See **Table 3**
* **Principal Actor**: User
* **Precondition**: GUI has control

|  |  |
| --- | --- |
| **Steps** | **Description** |
| Trigger | The User Starts running the program |
| Enter Parameters | The User enters parameters one by one. If any parameter is not entered, default values will be used if there are any. |

* **Name**: Run Session System Use case (SUC-2)
* **Brief Description**: See **Table 3**
* **Principal Actor**: User
* **Precondition**: All parameters are entered. Button “Start” is pressed.
* **Included Use Cases**: Validate Parameters

|  |  |
| --- | --- |
| **Steps** | **Description** |
| Trigger | The User pressed button “Start”. |
| Validation | Check that entered parameters are valid (numbers, parameter was not entered, etc.)  If parameters are not valid – show error message. |
| Prepare for the evolutionary process | Create initial population on the basis of a given Peach Pit file(each new citizen is a mutated variation of the ancestor) |
| Start evolutionary process | For every iteration repeat following steps:   * Calculate Fitness(includes running Peach Fuzzer Platform) * Sort population by fitness * Create child population on the basis of parameters chosen by user * Possibly mutate some of children to avoid convergence |
| Save results | Save results of each iteration in the xml file in accordance to the predefined format |

* **Name**: Return Output System Use case (SUC-3)
* **Brief Description**: See **Table 3**
* **Principal Actor**: User
* **Precondition**: Session finished properly
* **Extended Use Cases**: The user can see results both in the output window and save them in the files.

|  |  |
| --- | --- |
| **Steps** | **Description** |
| Trigger | The User pressed button “Show”. |
| Show statistics in a form of chart | The User can choose necessary parameter from a list and immediately the result in a form of the chart. Information is taken from xml file saved during the session. |
| Save shown statistics in the file | If the user wants he may save the results shown in the output window in a file. Path to save logs is selected by the user. |

# System Level Activity Diagram

The System Level Activity Diagram is presented on Figure 5 and Figure 6.

**Figure 5:** System Level Activity Diagram (Session Workflow)

|  |  |  |
| --- | --- | --- |
| **CUSTOMER** | **GUI** | **GA Module** |
| Enter Parameters  Run Session  Validate Parameters  Run Chosen GA Configuration  No  Yes  Show Error  Output results to xml file  Start Session Workflow  End Session Workflow |  |  |

**Figure 6:** System Level Activity Diagram (Statistics Workflow)

|  |  |  |
| --- | --- | --- |
| **CUSTOMER** | **GUI** | **Statistics Module** |
| Enter Parameters  Show Results in Output Window  No  Yes  Save results to the file  Start Statistics Workflow  End Statistics Workflow  Save results to file? |  |  |

# Class Diagram

**Figure 6**: Evolutionary Framework Class Diagram

1

\*

**GUIIface**

**Parameters**

**PitFileMode**

**GAFramework**

**Citizen**

<<uses>>>

<<uses>>>

<<uses>>>

**Statistics**

**CopyDir**

<<uses>>>

<<uses>>>

**MutationFunctions**

<<uses>>>

The system has 8 classes:

* **GUI Class:** GUIIface
  + Responsible for collecting data from the User, organizing it and sending to the Evolutionary class
* **Evolutionary Classes**: GAFramework, Citizen
  + Used for implementation of Genetic Algorithms(selection strategy, survival strategy, creating population , mutation, fitness analysis)
* **Supporting Classes**: PitFileMode, Parameters, CopyDir, Statistics and MutationFunctions.
  + . Supporting classes are used to help in calculations and data organization.

All of them are presented on **Figure 6**. Further are given detailed schemes of each class.

|  |
| --- |
| **GUIIface Class** |
| - FitnessStrategy : int  - FitnessStrategy\_name : String  - BadGenesChildren : int  - parentsChoiceStrategy : int  - parentsChoiceStrategy\_name : String  - elitrate : int  - survivalStrategy : int  - survivalStrategy\_name : String  - generate : int  - maxAge : int  - withScaling : bool  - mutationRate : int  - iterations : int  - PitFilePath : String  - logsPath : String  - PathOfPopulation : String  - Population : int  - PeachTestRange : String  - PDFPath : String  - PDFtime : String |
| - comboBox1\_SelectedIndexChanged(object, EventArgs) : void  - Start\_Button\_Click(object, EventArgs) : void  - Start\_Button\_2\_Click(object, EventArgs) : void  - BrowseButton1\_Click(object, EventArgs) :void  - BrowseButton2\_Click(object, EventArgs) : void  - BrowseButton3\_Click(object, EventArgs) : void  - DisableControls(Control) : void  - EnableControls(Control) : void  - paramCheck(Control) : void  - SetDefaultValues() : void  - FitnessChoicebutton\_Click(object, EventArgs) : void  - GAReturnbutton\_Click(object, EventArgs) : void  - FitnessStrategyComboBox\_SelectedIndexChange(object, EventArgs) : void  - GAApplybutton\_Click(object, EventArgs) : void  - applyGAvalues(object, EventArgs) : void  - SurvivalStrategycomboBox\_SelectedIndexChanged(object, EventArgs) : void  - ParentsChoicecomboBox\_SelectedIndexChanged(object, EventArgs) : void  - ScalingcheckBox\_CheckedChanged(object, EventArgs) : void  - HomeButton\_Click(object, EventArgs) : void  - StatButton\_Click(object, EventArgs) : void  - HomeButton2\_Click(object, EventArgs) : void  - PDFbutton\_Click(object, EventArgs) : void  - ChartcomboBox\_SelectedIndexChanged (object, EventArgs) |

|  |
| --- |
| **Parameters** |
| -LogsPath : String  -population : int  -PathOfPitPopulation :  String  -originalPitFile : String  -peachTestRange : String  -BadGenesChildren : int  -parentsChoiceStrategy : int  -elitrate : int  -survivalStrategy : int  -generate : int  -maxAge : int  -withScaling : bool  -PitFilePath : String  -mutationRate : double  -iterations : int |
| + setLogsPath(String) : void  + getLogsPath() : String  + setPopulation(int) : void  + getPopulation() : int  +setPathOfPitPopulation(String) : void  + getPathOfPitPopulation() : String  + setOriginalPitFile(String) : void  + getOriginalPitFile() : String  + setPeachTestrange(String) : void  + getPeachTestRange() : String  + setBadGeneschildren(int) : void  + getBadGenesChildren() : int  + setParentsChiceStrategy(int) : void  + getParentsChoiceStrategy() : int  + setElitrate(int) : void  + getElitrate() : int  + setSurvivalStrategy(int) : void  + getSurvivalStrategy() : int  + setGenrate(int) : void  + getGenrate() : int  + setMaxAge(int) : void  + getMaxAge() : int  + setWithScaling(bool ) : void  + getWithScaling() : bool  + setpitFilepath(String) : void  + getPitfilePath() : String  + setMutationRate(double) : void  + getMutationRate() : double  + setIterations(int) : void  + getIterations() : int |

|  |
| --- |
| **PtiFileMode** |
|  |
| +PitFileMode\_main(Parameters) : void |

|  |
| --- |
| **CopyDir** |
|  |
| +Copy (string, string): void  +CopyAll (DirectoryInfo, DirectoryInfo): void |

|  |
| --- |
| **Citizen** |
| -DMblocks : IEnumerable<XElement>  -Testblocks : IEnumerable<XElement>  -fitness : double  -age :int  -index : int  -faults :bool |
| + void setDMBlocks(IEnumerable<XElement>) : void  +getDMBlocks() : IEnumerable<XElement>  + setTestBlocks(IEnumerable<XElement> ) : void  + getTestBlocks() : IEnumerable<XElement>  + setFitness(double) : void  + getFitness() : double  + setIdx(int ) : void |

|  |
| --- |
| **Statistics** |
|  |
| + get\_statistics(List<Citizen>, String, int) : void  + check\_convergence(List<Citizen>, String) : int  + check\_invalidPits(List<Citizen>) : int  + readFile(String) : String  + get\_average(List<Citizen>) : double  + pitFilesWithFaults(List<Citizen>) : int |

|  |
| --- |
| **GAFramework** |
| + RAND\_MAX: int {readonly}  +String peach\_cmd : String {readonly}  -pathOfPit : String  -logsPath : String  -PeachTestRange : String  -OriginalPitFile : String  -data\_vector : Dictionary<string, string>  -data\_line : string  -pop\_list : List<Citizen>  -buffer : List<Citizen>  -rnd : Random  -BadGenesChildrenNumberToMake : int  -elitrate : int  -useGenotor : bool  -generate : int  -maxAge : int  -withScaling : bool  -GAmutate : double  -iterarions : int  -faults\_counter : int |
| +start(Parameters) : void  -calc\_fitness(String, List<Citizen>, String, String) : void  -init\_population(int, List<Citizen>) : void  -fitness\_sort(Citizen, Citizen) : int  -sort\_by\_fitness() : void  -mate(List<Citizen>, List<Citizen>, int, int, Boolean, int, int) : void  -crossover(List<Citizen>, int) : Citizen  -naive(List<Citizen>) : List<Citizen>  -fps(List<Citizen>) : List<Citizen>  -elitism(List<Citizen>, List<Citizen>, int) : void  - expScalling(List<Citizen>) : void  - aging(List<Citizen>, List<Citizen>, int) : void  - tournament(List<Citizen>, Boolean) : List<Citizen>  +CopyFolderContents(string, string) : void |

|  |
| --- |
| **MutationFunctions** |
|  |
| + mutate(XDocument, Random) : void  + flag\_(XElement, Random) : void  +choice\_(XElement, Random) : void  +number\_(XElement, Random) : void  +datamodel\_(XElement, Random) : void  +block\_(XElement, Random) : void  +blob\_(XElement, Random) : void  +string\_(XElement, Random) : void  +occurs(XElement, Random) : void  +signed(XElement, Random) : void  +endian(XElement, Random) : void  +size(XElement, Random, int) : void  +padCharacter(XElement, Random) : void  +null\_terminated(XElement) : void  +length(XElement, Random) : void  +minMaxOccurs(XElement, Random) : void  +mutable(XElement, Random) : void  +token(XElement, Random) : void  +value\_(XElement, Random): void  +crString(int, Random): String |