Generating Effective Test Suite for Multiparameter Software using ACTS Tool and its Verification using Code Coverage Tools

Abhinandan H. Patil, Krishnan Rangrajan

Abstract— Combinatorial testing is a practical method to test software with multiple input parameters. National Institute of Standards and Technology has developed tools which aid combinatorial testing. ACTS is one such tool which is freely available to users. In spite of this, very few software being developed are being tested systematically. In this paper we explore the effectiveness and suitability of ACTS tool to test software which has a multiparameter input. We chose a Java based software, College Time Table, a software which involves multiparameter input, as system under test. We could achieve 90% coverage of instructions, line, method and 100% class coverage with practical time and effort with ACTS tool. The process involved in getting above mentioned results is documented in this paper. Empirical data generated with the code coverage confirms the effectiveness of ACTS generated test suite for a simple project.

Index Terms— ACTS, CT, NIST, OpenClover, CTT, Eclipse, SDLC.

1 Introduction

In a typical Software Development Life Cycle (SDLC) 50-80% budget goes towards testing. In spite of this, 2003 National Institute of Standards and Technoloogy (NIST) report estimated a loss of 59.5 billion dollars to US economy due to inadequate testing [1]. In this light, software testing becomes critical. This is all the more critical as developers use pre-existing code which are now available as part of open source projects.

Several approaches exist to enable testing. One such field of testing is Combinatorial Testing (CT). A report by Nie indicates that CT is widely adopted by industry and researchers alike [2]. Exhaustive testing of all the possible combination of inputs and execution paths is a laborious task involving impractical man hours. CT is a method that can reduce cost and increase the effectiveness of testing. Pairwise testing is already in practice where 2-way combination of parameters is tested. However, as per NIST two way testing misses 10% to 40% or more system bugs [1]. Therefore higher level interaction testing is critical. New algorithms have made 4 way to 6 way testing possible. Some of the rare combination of inputs trigger the failures that would have escaped previous testing or extensive use. Such failures are known as interaction faillures. Traditional pairwise testing targets the 2-way interaction failures and has been in practice for quite some time. Till recently most tools would take impractically long time for generating the 3-way through 5-way arrays as the array generation process is mathematically complex. But the development of new algorithms recently has made the 3-way through 5-way array generation possible [3]. Two forms of combinatorial testing is possible: Configuration based combinatorial testing and Input parameter based combinatorial testing. A combination of both can also be used. Variety of software tools are available to assist CT. NIST gives few readily available tools for this purpose. They are

1) Automated Combinatorial Testing of Software (ACTS).
2) Combinatorial Coverage Measurement (CCM) tool.
3) Sequence Covering Array Generator.

In this paper we take ACTS tool [4] and develop a mechanism for its usage and show its effectiveness for the purpose of CT. CCM tool is useful for measuring the combinatorial coverage [5]. Sequence covering array generator is useful when sequences are involved. We successfully use this on a software which is free and available on SourceForge.

2 Brief Literature Survey Of CT

Much work has been done on estimating the fault detection effectiveness of CT [6]. The basic assumption of the CT is faults are deterministic in that failure triggering combination of input values always produce failures if it is present in the input. There much study being done on the tools and usage of CCM [7]. M F Johansen has used CCM to test software of industrial size in his PhD Thesis [8]. M F Johansen describes product line testing, which is the strategic testing of the product line to gain confidence for any configuration of it. CT has also been applied to industrial test suites [9]. Laleh et al. talk about input space modeling methodology, before applying CT to a system, the input space must be modeled [10]. CT approach is unique in each paper and is evident from few papers [11, 12]. P. Amman et al. talk about how to combine model chekers with specification based mutation to generate the test cases from formal software specification. Paolo et al. talk about validation of model and test tool generated test suites [12].

3 ACTS Tool

ACTS tool is a test generation tool. It generates t-way combinatorial tests sets. T in t-way can range from 1 through 6. A system in ACTS tool is specified by a set of parameters and their values. Given any t parameters (out of all the parameters) of system, every combination of values of these t parameters is covered atleast one test in the test set.

ACTS makes use of several algorithms for the test generation. The algorithms include IPOG, IPOG-D, IPOG-F and IPOG-F2.

IPOG-D is preferred for larger systems while IPOG, IPOG-F, IPOG-F2 work best for moderate size system.

4 Open Clover

OpenClover is an open source code coverage tool[13]. Code coverage is a quantitative data about code covered as a part of test execution. It shows which part of the code is tested and which is not tested. Typically, tester does the code coverage in an iterative manner till the required criteria is met. Code coverage is done for the following reason:

- To know whether testing is adequate.
- To maintain the quality of code.

Code coverage is not a replacement to code review and good programming practice but it complements the coding activity. There exist three types of coverage tools. They are:

- Source code instrumentation tools
- Byte code instrumentation tools
- Runtime information collecting tools.

Clover uses the source code instrumentating as the source code instrumenting produces more accurate results. Types of coverage measured by various tools are:

- Statement coverage
- Branch coverage
- Method coverage.

Clover combines the above mentioned criteria viz. statement coverage, branch coverage and method coverage criteria to arrive at Total Percentage of Coverage (TPC).
TPC is calculated as follows:

$$TPC = (BT + BF + SC + MC)/(2 \times B + S + M) \times 100\%$$

where

BT: Branches that evaluated to "true" at least once
BF: Branches that evaluated to "false" at least once
SC: Statements covered
MC: Methods entered

B: Total number of branches
S: Total number of statements
M: Total number of methods

Clover actually sees the real code structure and uses the source code instrumentation. Only line coverage is possible with byte code instrumentation tools. However, statement coverage is possible with Clover as it uses the source code instrumentation.

4 College Time Table

College Time Table (CTT) software is a simple Java based tool for generating small sized school or college time table [14]. CTT utility collects the information on the fly without expecting the details such as number of teachers, their name, subjects etc.

CTT is picked for demonstrating the ACTS tool usage since:

1) Practical sized code
2) Combinations of parameters are used as input and therefore it is a suitable software to demonstrate the functionality of ACTS.
3) We wanted to demonstrate the applicability of the method proposed by us on a software which we had not much detailed knowledge about.

5 Method Involved In Generating The Test Cases From ACTS Tool And Gathering The Coverage Data

Table 1. Parameters and their values in ACTS for CTT software

Parameter	**Parameter values**
File_Operation	New_Time_Table, Save_Time_Table, Save_Time_Table_As, Load_Time_Table, Null
Print_Operation	Print_Current, Print_All_Individuals, Print_All_Classes, Print_Master_Table
Load_Demo_Time_Table	Demo_Time_Table, Null
Time_Table_Operation	Printer, Global_Counts, Remove_Gaps_Doubles, Freeze_Cell, Multi_Freeze, Clear_Freeze, Find, Next_Find, Find_And_Replace, Wizard-01, Insert_Row, Swap_Time_Table, Wizard-02, Delete_Row
Constraints: (File_Operation != "Null") => (Load_Demo_Time_Table == "Null") (Load_Demo_Time_Table != "Null") => (File_Operation == "Null")	

ACTS tool needed for this work was downloaded from the NIST website. CTT software which involves multiple input parametrs and their combinations was taken from the github. The target SUT (CTT in this case) was imported in Integrated Development Environment (IDE) which was Eclipse. Once chosen SUT is imported, next step is installation of the code coverage plugin tool. OpenClover plugins are available for many popular IDEs. The SUT (CTT) is instrumented and built. Next step is to launch the SUT as usual.

5.1 ACTS tool usage for generating the test cases

ACTS tool comes with user guide. User guide contains information on how the tool needs to be used. ACTS tool was launched. The parameters and parameter values along with relations and constraints if any need to be populated in system under ACTS tool. Once these are populated the system can be built. Building the system is making ACTS tool generate the test cases. Each row of the output is one test case. To begin with certain combination of parameter and parameter values can be chosen along with constraints and relations. All the test cases in ACTS tool can be run. At the end of run of all the test cases, Clover would have collected the data in the background. The open Clover data can be exported in various output forms. In this case html format was chosen. From the output one can infer the data such as:

- Instruction coverage
- Branch coverage
- Condition coverage
- Line coverage
- Method coverage
- Class coverage

If the required criteria of code coverage is not met, the parameter modeling the ACTS tool can be revisted and refined. It is a iterative process and can be repeated till the required coverage criteria is met.

Figure 1 summarizes the steps mentioned for this work.

Table 1. gives the final set of parameter and parameter values along with constraints chosen for this work. The iterative work was concluded for 90% instruction coverage. Results section discusses the results in detail.

Although the steps mentioned in section 5 and subsection 5.1 are specific for a given software (CTT), for a given IDE (Eclipse) and code coverage tool (OpenClover), the steps involved are generic in nature and can be used for other software where combinations are involved.

Following configuration was chosen:

- Algorithm used: IPOG
- Strength chosen: 2
- Mode chosen: Scratch
- Constraint handling: CSP Solver

ACTS output statistics are as follows:

- Number of test cases: 71
- Number of covered combinations: 188

For the above mentioned configuration, ACTS took mere 0.094 seconds to generate the output to be exported in various formats.

6 Results And Results Analysis

Table 2 summarizes the results of test execution for CTT project.

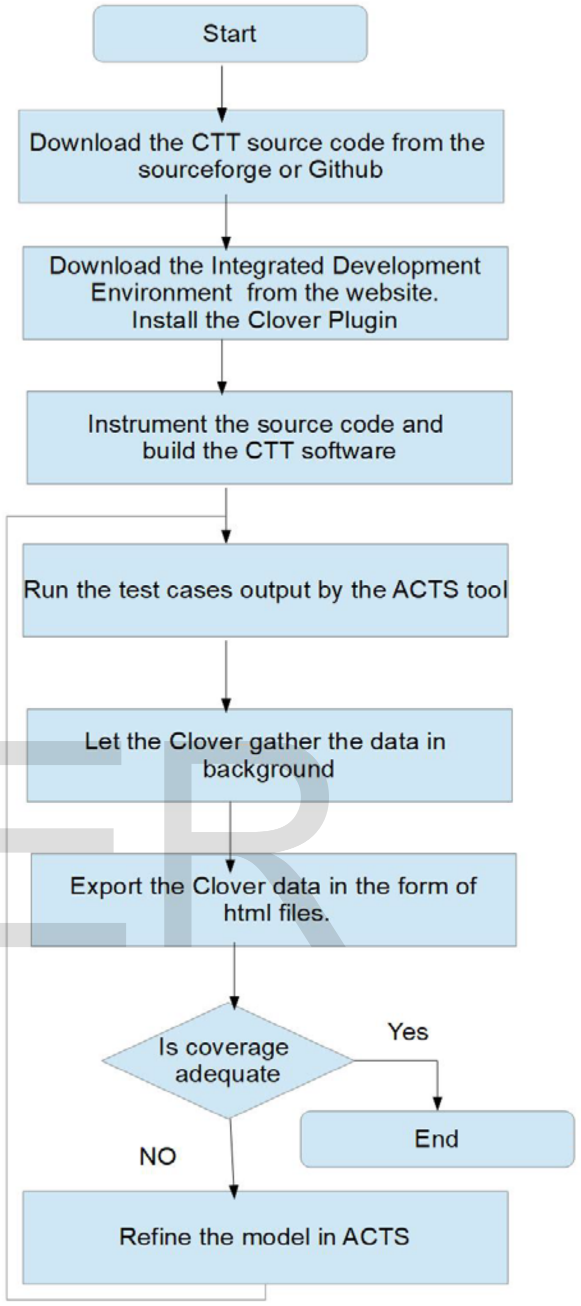

Start
Download the CTT source code from the sourceforge or Github
Download the Integrated Development Environment from the website. Install the Clover Plugin
Instrument the source code and build the CTT software
Run the test cases output by the ACTS tool
Let the Clover gather the data in background
Export the Clover data in the form of html files.
Is coverage adequate
Yes
End
NO
Refine the model in ACTS

Figure 1. Process flow diagram for generating the ACTS test suite for CTT software and measuring the coverage

Table 2. Clover coverage data for CTT software

CTT Project

- *Abhinandan H. Patil is currently pursuing Doctor of Philosophy program in CS and IS Department in BITS Pilani, Goa Campus, University, India, Mob-919886406214. E-mail: Abhinandan_patil_1414@yahoo.com*
- *Neena Goveas is currently Professor in CS and IS department in BITS Pilani, Goa, University, India, E-mail: Neena@goa.bits-pilani.ac.in*
- Krishnan Rangarajan is Professor in CSE department in Dayanand Sagar College of Engineering, Bengaluru, India Email: Krishnanr1234@gmail.com

Entity	Total	Missed	Covered
Instructions	13483	1080	91%
Branches	784	195	75%
Lines	2383	239	90%
Methods	343	40	89%
Classes	89	0	100%

Appendix B gives the visual output for the table. Detailed logs are kept in Google docs repository [15] and they act as supplementary information. As can be seen from the Table 2, class coverage was 100% while lines, instruction and method coverage was around 90%. Branch coverage was around 75% for the 70 test cases. Further, only success path in the code was tested and the error scenarios were not explored. In addition, some dead code existed in the CTT project. As mentioned earlier, since the vital entities were around 90% (Except branch coverage), the refining of parameter model in the ACTS was concluded.

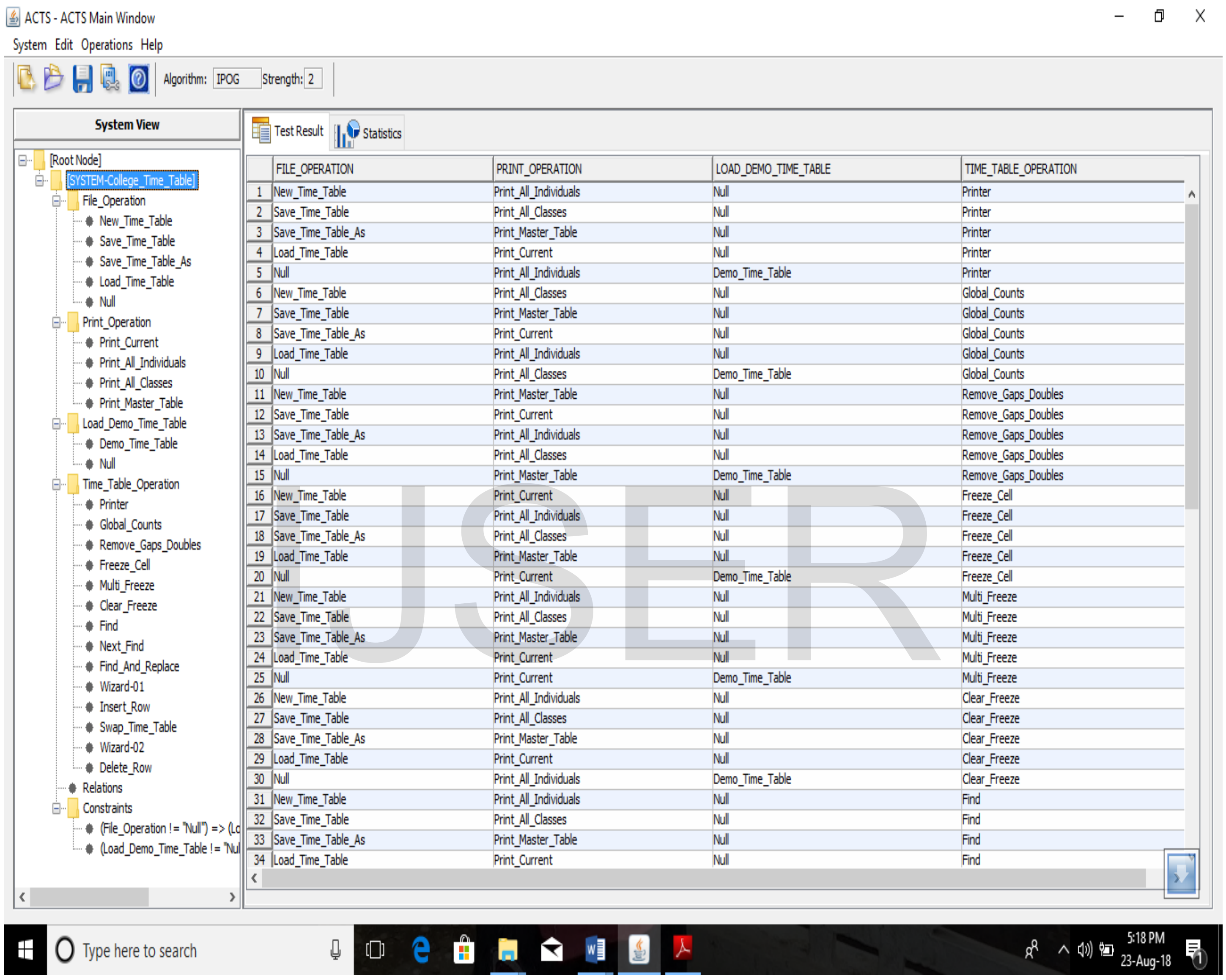
7 Conclusion

In this paper we present an implementation of use of combinatorial testing based tool for generation of test suite. We have used tools provided by NIST, ACTS and OpenClover. We use Clover a tool for for measuring the code coverage. We document the findings of generating test suite for multi parameter software and its verification using code coverage tools. We use as a SUT a multiparameter Java based software. We find that the implementation can be used on any mutiparameter software. The effectiveness of the ACTS tool generated test suite is cross verified with traditional metrics code coverage. The documented process in this paper could achieve 90% coverage for the vital entities of coverage metrics with practical time and effort. The process documented in this paper could be effectively used for any softwares which involve combinations of input parameters. Since several free software are used by users with their modifications incorporates, the method detailed in this work provides an effective way of testing.

References

[1] Y. L. D.Richard Kuhn, Raghu N. Kacker, "Practical Combinatorial Testing." http://csrc.nist.gov/groups/SNS/acts/documents/SP800-142-101006.pdf, 2010.
[2] C.Nie, "A survey of combinatorial testing," ACM Computing Surveys, vol. 43, no. 2, 2011.
[3] Y. L. D.Richard Kuhn, Raghu N.Kacker, Introduction to Combinatorial Testing. A Chapman and Hall Books, 2013.
[4] "ACTS tool website." http://csrc.nist.gov/groups/SNS/acts/index.html.
[5] "CCM tool website." http://csrc.nist.gov/groups/SNS/acts/index.html.
[6] Y. L. D.Richard Kuhn, Raghu Kacker, "Estimating fault detection effectiveness," IEEE International Conference on Software Testing, Verification, and Validation Workshops, 2014.
[7] Y. L. D.Richard Kuhn, Raghu Kacker, "Combinatorial coverage measurement concepts and applications," International Workshop on Combinatorial Testing, 2013.
[8] M. F. Johansen, Testing Product Lines of Industrial Size: Advacements in Combinatorial Interaction Testing. PhD thesis, PhD thesis, University of Oslo, 2013.
[9] Y. L. Laleh Shikh Gholamhossein Ghandehari, Mehra N. Bourazjany, "Applying combinatorial testing to the siemens suite," International Workshop on Combinatorial Testing, 2013.
[10] Y. L. Laleh Shikh Gholamhossein Ghandehari, Mehra N. Bourazjany, "An input space modeling methodology for combinatorial testing," International Workshop on Combinatorial Testing,2013.
[11] P.Amman and P.E.Black, "Abstracting formal specifications to generate software tests via model checking," Proc. 18th Digital Avionics System Conference, vol. 2, no. 10.A.6, pp. 1–10.
[12] P. V. Paolo Arcaini, Angelo Gargantini, "Validation of models and tests for constrained combinatorial interaction testing," IEEE International Conference on Software Testing, Verification,and Validation Workshops, 2014.
[13] "OpenClover website.", http://openclover.org.
[14] "College Time Table website.", https://sourceforge.net/projects/collegetimetable/
[15] "CTT test execution log files", https://drive.google.com/drive/u/1/folders/1t_fT6rd3OLTfLiipmRyQi9 9_436emGAk
[16] "Eclipse website.",https://www.eclipse.org/downloads/

APPENDIX A: ACTS TOOL POPULATED DATA FOR CTT



ACTS - ACTS Main Window

System Edit Operations Help

Algorithm: IPOG Strength: 2

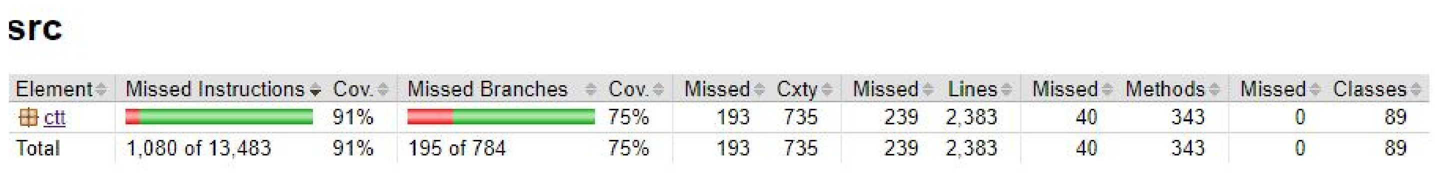
System View

- [Root Node]
  - [SYSTEM-College_Time_Table]
    - File_Operation
      - New_Time_Table
      - Save_Time_Table
      - Save_Time_Table_As
      - Load_Time_Table
      - Null
    - Print_Operation
      - Print_Current
      - Print_All_Individuals
      - Print_All_Classes
      - Print_Master_Table
    - Load_Demo_Time_Table
      - Demo_Time_Table
      - Null
    - Time_Table_Operation
      - Printer
      - Global_Counts
      - Remove_Gaps_Doubles
      - Freeze_Cell
      - Multi_Freeze
      - Clear_Freeze
      - Find
      - Next_Find
      - Find_And_Replace
      - Wizard-01
      - Insert_Row
      - Swap_Time_Table
      - Wizard-02
      - Delete_Row
    - Relations
    - Constraints
      - (File_Operation != "Null") => (Lo
      - (Load_Demo_Time_Table != "Nul

Test Result | Statistics

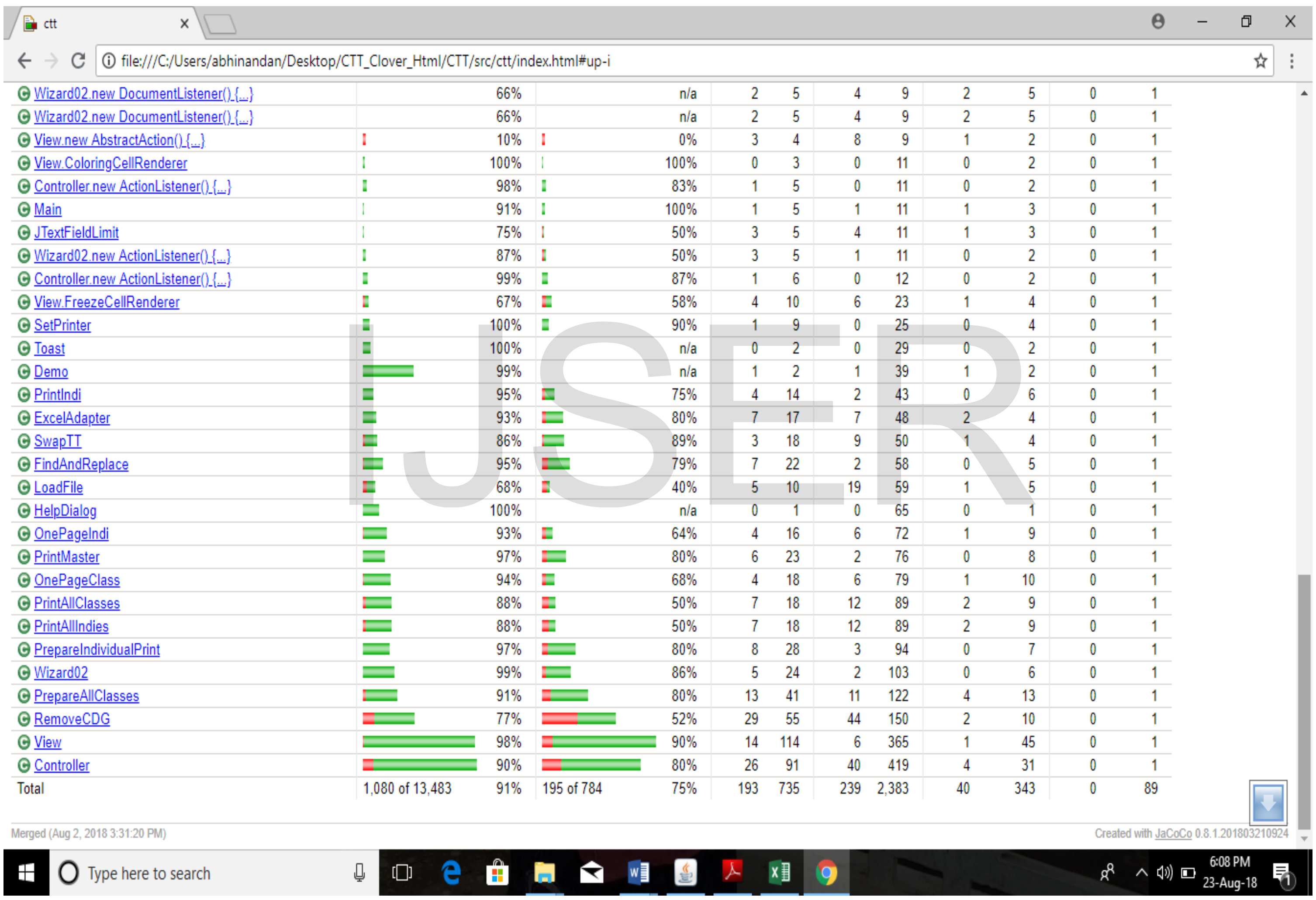
	FILE_OPERATION	PRINT_OPERATION	LOAD_DEMO_TIME_TABLE	TIME_TABLE_OPERATION
1	New_Time_Table	Print_All_Individuals	Null	Printer
2	Save_Time_Table	Print_All_Classes	Null	Printer
3	Save_Time_Table_As	Print_Master_Table	Null	Printer
4	Load_Time_Table	Print_Current	Null	Printer
5	Null	Print_All_Individuals	Demo_Time_Table	Printer
6	New_Time_Table	Print_All_Classes	Null	Global_Counts
7	Save_Time_Table	Print_Master_Table	Null	Global_Counts
8	Save_Time_Table_As	Print_Current	Null	Global_Counts
9	Load_Time_Table	Print_All_Individuals	Null	Global_Counts
10	Null	Print_All_Classes	Demo_Time_Table	Global_Counts
11	New_Time_Table	Print_Master_Table	Null	Remove_Gaps_Doubles
12	Save_Time_Table	Print_Current	Null	Remove_Gaps_Doubles
13	Save_Time_Table_As	Print_All_Individuals	Null	Remove_Gaps_Doubles
14	Load_Time_Table	Print_All_Classes	Null	Remove_Gaps_Doubles
15	Null	Print_Master_Table	Demo_Time_Table	Remove_Gaps_Doubles
16	New_Time_Table	Print_Current	Null	Freeze_Cell
17	Save_Time_Table	Print_All_Individuals	Null	Freeze_Cell
18	Save_Time_Table_As	Print_All_Classes	Null	Freeze_Cell
19	Load_Time_Table	Print_Master_Table	Null	Freeze_Cell
20	Null	Print_Current	Demo_Time_Table	Freeze_Cell
21	New_Time_Table	Print_All_Individuals	Null	Multi_Freeze
22	Save_Time_Table	Print_All_Classes	Null	Multi_Freeze
23	Save_Time_Table_As	Print_Master_Table	Null	Multi_Freeze
24	Load_Time_Table	Print_Current	Null	Multi_Freeze
25	Null	Print_Current	Demo_Time_Table	Multi_Freeze
26	New_Time_Table	Print_All_Individuals	Null	Clear_Freeze
27	Save_Time_Table	Print_All_Classes	Null	Clear_Freeze
28	Save_Time_Table_As	Print_Master_Table	Null	Clear_Freeze
29	Load_Time_Table	Print_Current	Null	Clear_Freeze
30	Null	Print_All_Individuals	Demo_Time_Table	Clear_Freeze
31	New_Time_Table	Print_All_Individuals	Null	Find
32	Save_Time_Table	Print_All_Classes	Null	Find
33	Save_Time_Table_As	Print_Master_Table	Null	Find
34	Load_Time_Table	Print_Current	Null	Find

Appendix B: Open Clover Output Window



src

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods	Missed	Classes
ctt		91%		75%	193	735	239	2,383	40	343	0	89
Total	1,080 of 13,483	91%	195 of 784	75%	193	735	239	2,383	40	343	0	89



ctt

file:///C:/Users/abhinandan/Desktop/CTT_Clover_Html/CTT/src/ctt/index.html#up-i

Wizard02.new DocumentListener() {...}		66%		n/a	2	5	4	9	2	5	0	1
Wizard02.new DocumentListener() {...}		66%		n/a	2	5	4	9	2	5	0	1
View.new AbstractAction() {...}		10%		0%	3	4	8	9	1	2	0	1
View.ColoringCellRenderer		100%		100%	0	3	0	11	0	2	0	1
Controller.new ActionListener() {...}		98%		83%	1	5	0	11	0	2	0	1
Main		91%		100%	1	5	1	11	1	3	0	1
JTextFieldLimit		75%		50%	3	5	4	11	1	3	0	1
Wizard02.new ActionListener() {...}		87%		50%	3	5	1	11	0	2	0	1
Controller.new ActionListener() {...}		99%		87%	1	6	0	12	0	2	0	1
View.FreezeCellRenderer		67%		58%	4	10	6	23	1	4	0	1
SetPrinter		100%		90%	1	9	0	25	0	4	0	1
Toast		100%		n/a	0	2	0	29	0	2	0	1
Demo		99%		n/a	1	2	1	39	1	2	0	1
PrintIndi		95%		75%	4	14	2	43	0	6	0	1
ExcelAdapter		93%		80%	7	17	7	48	2	4	0	1
SwapTT		86%		89%	3	18	9	50	1	4	0	1
FindAndReplace		95%		79%	7	22	2	58	0	5	0	1
LoadFile		68%		40%	5	10	19	59	1	5	0	1
HelpDialog		100%		n/a	0	1	0	65	0	1	0	1
OnePageIndi		93%		64%	4	16	6	72	1	9	0	1
PrintMaster		97%		80%	6	23	2	76	0	8	0	1
OnePageClass		94%		68%	4	18	6	79	1	10	0	1
PrintAllClasses		88%		50%	7	18	12	89	2	9	0	1
PrintAllIndies		88%		50%	7	18	12	89	2	9	0	1
PrepareIndividualPrint		97%		80%	8	28	3	94	0	7	0	1
Wizard02		99%		86%	5	24	2	103	0	6	0	1
PrepareAllClasses		91%		80%	13	41	11	122	4	13	0	1
RemoveCDG		77%		52%	29	55	44	150	2	10	0	1
View		98%		90%	14	114	6	365	1	45	0	1
Controller		90%		80%	26	91	40	419	4	31	0	1
Total	1,080 of 13,483	91%	195 of 784	75%	193	735	239	2,383	40	343	0	89

Merged (Aug 2, 2018 3:31:20 PM)

Created with JaCoCo 0.8.1.201803210924

Type here to search

6:08 PM
23-Aug-18