CART: Context-Aware Regression Testing

Yizhen Chen and Mei-Hwa Chen

***Abstract*— Regression testing is an essential process for evolving software systems. Existing techniques focus on the impact of code changes on the program behavior. However, changes in the execution context, such as library, database, APIs, and hosting servers, can also affect the behavior of the software. This paper presents a new selective regression approach that not only accounts for modification of the code, but also for the changes in the execution context. Context-Aware Regression Testing uses the state of the pre-conditions of a function to determine if the function is affected by the change, and selects tests that execute the affected functions for regression testing. The preconditions are identified by using dynamic likely invariants that preserve the properties required for correct execution of the function. In the meantime, these invariants carry context information that the function needs to be executed in. When a change takes place regardless of whether it is in the code or in the execution context, some invariants may be affected by the change that consequently affects the precondition of a function and deviates the function from its abnormal behavior. Thus, the test cases executing this function should be re-tested to investigate its behavior after the change. Our empirical studies show that the context-aware regression testing effectively selected all the fault-revealing test cases. The results suggest that the technique is safe and effective.**

***Index Terms*—selective regression testing, program likely invariant, dynamic analysis**

I. Introduction

Most software applications are continuous evolving, consequently, corrective, perfective, and adaptive maintenance activities need to be frequently performed on production software. Each of these activities will make changes to the software. In addition to create new tests for the new and/or changed features, regression testing must be performed to investigate if the changes adversely introduce regression faults. Furthermore, for multi-team software development, each team develops a set of features in its own environment, and the features are then merged in the deployment context. The changes of the execution environment such as libraries, DBMS, servers, operating systems, can also affect the behaviors of the software. Although there might be no changes in the code, regression testing must be performed to check if the software works well in the new execution environment. Because of the frequent evolving nature and the new tests are continuously added to the regression test suite, the cost of the regression testing is increasingly expensive. Thus, an effective selection from the regression test suite to perform efficient regression testing is essential to ensure that any change will not disrupt or delay the service of the production software.

A number of selective regression testing techniques have been proposed [ ], most of them focus on selecting modification-traversing tests that execute the changed part of the program. The techniques that based on dataflow analysis select tests that execute the new, deleted, and modified definition-use pairs of variables [ ]. The graph-based approaches compare the dependence graphs (CDG, SDG, or PDG) of the original and the modified programs, and select the tests that exercise the mismatched nodes in the graphs [ ]. The program slicing based approaches select tests that exercise the differences between the slices (dynamitic or SDC). The firewall approaches select tests that based on the nature of the interaction between the changed modules [ ].

These existing approaches mainly focus on selecting tests that exercise the changed part of the program, but they do not consider unchanged but affected parts, and the changes in the execution environment that affect the program behaviors. In [ ], one of the assumptions for a safe regression testing is that when a modified program P' is tested with a test case t, all factors that might influence the output of P', except for the code in P', are kept constant with respect to their states when the original program P was tested with t. This assumption cannot be held if the execution environment is changed and can potentially affect the behavior of the modified program P'. Thus, most of the existing techniques will not be safe when the changes take place in the execution environment. In [ ], the results of their empirical studies suggest that changes in the execution environments including libraries and APIs most likely would affect programs' behaviors.

This paper presents a safe selective regression testing technique, Context-Aware Regression Testing (CART), that selects tests executed affected functions regardless changed or unchanged. The change ripple effects on the unchanged functions can stem from the indirect dependences on a changed function, for example, an unchanged function affected via the dependence on an instance attribute changed by a modified function. They can also be propagated from the changes in the execution context such as library, databases, or containers, etc. In this paper, we focus on library and database changes that often take place in multi-team development environment. These change ripple effects can cause the program to fail on previously successful tests, which most likely will not be re-tested by the existing approaches, because they do not exercise any changes of the modified program.

Our regression test selection criterion is to select test cases whose execution traces include at least one program property that preserved for all the successful executions and is affected due to the changes. We use program invariants as means for capturing persistent program properties that should hold for

successful executions. Program invariants have been suggested by a number of studies that they can be used to investigate program behaviors [ ]. A program invariant is a condition that holds the program's property at a given point, where the invariant should hold a true value if the program's execution behaves as expected; otherwise, a false value should be observed. To collect program invariants, we use Daikon in a training phase to identify likely dynamic program invariants. The Daikon invariant detector, developed by Ernst et al. [], has been used in many research areas of software engineering, such as for monitoring runtime heap and memory for security protection [], inferring program contracts [], automated software fault localization [], identifying non-equivalent mutants in mutation testing [], verifying component-based software systems [], detecting dynamic invariants of relational databases and their applications [], software reliability prediction []. In our previous study [ ], we used selected likely invariants to monitor and detect online software anomalous behaviors. Daikon detects dynamic invariants by instrumenting subject programs at selected points to trace the values of the variables at these program points during the executions on a test suite and infers invariants over both instrumented and derived variables.

We use Daikon to instrument the subject software and run the regression test suite to create an Invariant Traceability Matrix (ITM), where each row denotes the occurrence of an invariant in a test case and the program point of the invariant. These invariants also indicate which library calls and entity objects the invariants are associated with. This matrix can also be created during the testing process, prior to the regression testing phase. After each change, if it is a code change then we run Daikon with the modified program and compare the invariants sets of the programs before and after the changes. The differences between the two invariants sets are considered affected invariants. If it is an environment change, then the invariants associated with the changed library calls or database entities are considered affected. Our algorithm selects the test cases that contains the affected invariants for regression testing.

The contributions of this paper are:

1. The impact of the changes in the execution context on the program behavior is difficult to identify and resolve in software maintenance. As to our best knowledge, the existing techniques for regression testing have barely addressed this issue. Most of the safe selection approaches assume that the factors that are external to the subject program are remain constant, which is often impractical for the modern multi-team developed evolving software. This paper presents a new approach addressing this challenging issue and the proposed technique can effectively select the test cases relevant to the changed environment.
2. Our approach is fully automated and we have implemented a prototype to conduct empirical studies. The results show that CART significantly reduced the size of the regression test suite while selected all the fault-revealing regression tests. Thus, regression testing can be performed efficiently and effectively to maintain the quality of the software after changes.

The remainder of this paper is organized as the following: the proposed approach is described in Section 2. In Section 3 we present our empirical studies. Section 4 gives an overview of the related techniques. The conclusions and the future work are given in Section 5.

II. METHODOLOGY

The basic unit of the investigation in our model is function (method in object-oriented software), and the points of interest are the invariants in the pre- and postconditions of each function. The precondition of a function specifies the state of the program that must be held before the invocation of the function to ensure that the function can be correctly executed. The postcondition of a function specifies the state of the program that must be true after the execution of the function. Each precondition and postcondition contains zero or more program invariants.

Given a program *P*, the modified program P', a regression test suite *T* and the execution context *EC* (including libraries, APIs, databases), our selective regression testing is described as follows:

1. If a function *f* in *P* is deleted in *P'*, then all the tests exercising *f* become obsolete, and should be removed from T.
2. If an attribute (a variable) in *P* is changed in *P'*, then all the functions that use this changed attribute (variable) are affected. The test cases exercising these functions will need to be retested.
3. If a function *f* is modified; then a new test *t* will be created to test the modified function *f'*. In a corrective activity, when a failure is encountered in a test case *t*, the faulty function(s) are modified until it passes *t*.
   a. All the tests in *T* that exercise *f* and their postconditions of *f* are different from the postcondition of *f'* in *t* will need to be retested, and the postconditions will be updated.
   b. Every function *f''* in *P* that is the same in *P'*, but its precondition and/or postcondition is affected by the postconditions of *f'*, then *f''* is affected. All the tests that exercise *f''* will be retested.
4. A new function *g* is added to *P'*, then new tests will be created to test *g*. Some functions in *P* will be modified in *P'* to call *g*; all the tests in *T* that exercise these functions and their callers will need to be tested. This is the same as (3).
5. A function *l* library *L* in *EC* is changed, then all the functions in *P* that call *l* are affected, and all the tests exercising these functions need to be retested. Note that if the changes to *L* are unknown, we take a conservative way that considers all the functions making library calls affected, and select all the tests exercising these functions to retest.
6. A database schema is changed, then all the entity objects associated with the changed table are affected, and all the tests exercising these objects need to be retested.

To obtain preconditions and postconditions for each function invocation, we use Daikon to detect program invariants at before and after each function call by executing a large number of test cases. Then we creates an invariant traceability matrix (ITM) where each column records the appearance of the invariants in the precondition and postcondition of each function executed in a test case, and each row indicates the test cases in which an invariant occurs. Figure 1 shows the algorithm for the test case selection.

Algorithm : Test Case Selection

```
Input :
    T = {t0, t1, .., tn}: a test suite of P
    ITM[][]: Invariant Traceability Matrix of P w.r.t. T
    INV: a set of affected invariants in P
    F: a set of modified functions
    L:  a set of modified library functions
    D: a set of modified tables
Output:
    T':  a set of selected test cases
T' = Ø
 i = 0;
While (i < |T|)  {
    for each inv in INV {
        if (ITM[inv.index][i] ==1) {
            T'=T'U{ti};
i++;
        }
    }
    for each f in F{
        if (ITM[f.index][i]==1) {
            T'=T'U{ti};
                i++
        }
    }
      for each l in L and inv is associated with l{
        if ITM[inv.index][i] == 1)
            T'=T'U{ti}
            i++;
        }
       for each d in D; obj is associated with d, and inv
          is associated with obj {
        if ITM[inv.index][i] == 1)
            T'=T'U{ti}
            i++;
        }
}
return T'
}
```

Figure 1. The selective regression algorithm.

Program	Validation	JXPath	Lang	BookStore
Classes	52	156	81	36
Functions	1,200	1960	3,457	255
LOC	15,703	24,293	37,545	1,971
Test Cases	487	609	3,800	238

III. EMPIRICAL STUDY

We have developed a prototype of CART and conducted empirical studies on four open source software: Lang, Jxpath, and Validation are obtained from Apache Software Foundation, and BookStore is from GitHub. Table 1 shows the details of the software. Apache Software Foundation provides delicate issue track reports and test suites for each project. We selected a set of reported faults and restored them to the programs, and then removed them from the programs one at time during our study. GitHub provides online project hosting services. Many open source projects are hosted on GitHub, and GitHub provides issue tracking services. The first study is to demonstrate selective regression testing after a corrective activity. The second study is to show when a dependent library is changed, how CART select regression tests to re-test. The third study presents the case where a change was made to a database, how the program was affected and how CART selected regression tests.

These studies were conducted in two phases. The first phase is to observe the relationships between the likely invariants and the test cases. Daikon was used to instrument the subject software and to run the regression test suite to construct the invariant traceability matrix (ITM). This phase can be done during the testing process and incrementally update ITM when the test suite is updated. The second phase is conducted when a maintenance activity takes place. Details of these studies are given in the following sections.

A. *Empirical Study 1*

In the first study, Lang and JXPath from Apache Software Foundation were used. Apache Common Lang project is a complement to standard java libraries. It provides some extra functions for manipulating the core classes of standard java libraries. In the Lang project, we selected five faults related to the program statement changes and 3475 test cases. JXPATH is an interpreter for an expression language XPath and applies XPath to graph all kinds of objects. We also selected five faults related to statement changes and 609 test cases. The selected faults were restored to the programs and the following regression testing process was performed.

Step 1: Run Daikon on the faulty program P and the test suite T to create the ITM. When a failure is encountered in tf, the associated fault is removed from P, and the invariants obtained from this execution, I are recorded.

Step 2: Run Daikon on the modified P with tf , recording the invariants I', and update ITM.

Step 3: Compare I and I', and the resulting difference set Iaffect is the set of affected invariants.

Step 4: Use algorithm to select test cases from T and retest the modified P with the selected test cases.

Step 5: Repeat this process until all the faults are removed.

Versions	V1	V2	V3	V4	V5
classes	1	1	2	1	2
functions	1	1	8	6	10
statements	5	1	94	50	20
CART selected	9 (0.23)	43 (1.13%)	36 (0.94%)	33 (0.87%)	33 (0.87%)
ET selected	9 (0.23)	43 (1.13%)	36 (0.94%)	33 (0.87%)	33 (0.87%)

The results of the regression testing on Lang are summarized in Table 1. For all the five correctives, CART performed the same as the execution trace based approach and only selected around 1% of the tests, which significantly reduced the maintenance effort.

The first column denotes the version of the program . For each version of the program we modified the program to fix failed test cases. The number of class modified and modified functions, lines of code to be changed are listed in the second and third columns. The number of test cases selected by our method is listed in fourth column; the test cases selected by using execution trace method are listed on the fifth column. Compared with retest all, our method has the effect of saving 99% on average.

In this experiment, test cases selected by our method are equivalent to the test cases selected by using the execution trace method. The modification made to the program is to add a predictor or change entire function or use a different data structure. All the test cases executing these functions will execute the modified statement. This means that the execution trace method is equivalent to the one finding test cases executing the modified function. Besides, some modification has very limited impact on a program. For example, in version 1, only 1 function and 5 lines of code are changed compared to 3,457 functions and 37,545 lines of code; it is a very small change and it reordered the statements and removed the empty space in a return value. Such changes will not affect on program invariants changes. In this experiment, many test cases are selected by finding test cases executing modified function .

Versions	V1	V2	V3	V4	V5
m-classes	5	3	2	2	2
m-functions	16	13	2	2	2
m-statements	194	94	8	25	127
CART	3 (0.49%)	10 (1.64%)	42 (6.89%)	59 (9.68%)	301 (49.42%)
ET	3 (0.49%)	10 (1.64%)	32 (5.25%)	40 (6.56%)	271 (44.49%)

Table 2 summarizes the results of JXPath. In version 1, one class and four of its subclasses were modified to correct a fault. These modified classes are loosely coupled with the other classes, and their changes have very minor impact on the other classes. Therefore, even though this version had the largest number of statement changes, it only needed to rerun 0.49% of the test cases. In version 5, the modifications were made to a static initialization block in class JXPathContextReferenceImpl and every time this class was loaded this static initialization block will be executed. In addition, this class is a core class of JXPath, which is an implementation of JXPathContext that include many functions, and many test cases refer to this class. This is reason why so many test cases were selected.

In summary, in these two experiments our approach is as safe as the Execution Trace based approach, when the modifications are made to the attributes, our approach select more test cases as execution trace does, but includes test cases selected by execution trace. In this case, our approach is safer because it selected fault-revealing tests cases that were not selected by the Execution Trace based. However, because our approach is based on the function level not on the statement level, so when a test exercising a non-modified path in a modified function, then this test will be selected by our approach but not by the Execution trace based approach, thus our approach, in general, will select more regression tests.

B. *Empirical Study II*

The second study focused on changes on the program's dependent libraries. When a library is changed, in most cases it is difficult to identify where the changes are and how the program will be affected. Our approach selects all the tests that use the library. In our experiment, Daikon generated program invariants based on every statement executed, including calls to library functions, so for each test case, we know which library calls were used.

We used Validation and JXPath from Apache Software Foundation. Both programs are maven projects , and the dependent libraries can be easily changed by modifying the pom file. In the Validation program we downgraded three versions of its dependent libraries, and for Jxpath we downgraded two versions of its dependent libraries. Finding that many libraries are backward compatible but not forward compatible, we chose to downgrade to the older versions of the library to detect the regression faults that are difficult to find. Table 3 shows the results of Validation, and Table 4 shows the results of JXPath. Both experiments show that all the affected test cases (failed in the changed library) were selected by our approach; thus, it is safe with respect to the library changes. Both experiments obtained high recall and precision. This is because the changes in the libraries are significant; if the two versions of the library are very similar, then the precision may be lower, but the recall will still be high.

Changed Library	commons-beanutils	commons-digester	commons-collections
Modification	V1.92-> V1.2	V1.8.1 -> V1.3	V3.2 -> V2.0
Affected Test Cases	99	99	101
Selected Test Cases	103	103	105
Recall	1	1	1
Precision	96.11%	96.11%	96.19%

	commons-beanutils	jdom
Modification	V1.7.0-> V1.2	V2.02 -> V1.1
Affected Test Cases	118	60
Selected Test Cases	118	60
Recall	1	1
Precision	1	1

C. *Empirical Study III*

Changed Library	V1	**V2**	**V3**

Modification	Modify book table's column name from name to title	Delete table caegory	Add column status in book table
Affected test cases	3	12	0
Selected test cases	31	62	31
Recall	1	1	NA
Precision	9.68%	19.35%	0

The third study focused on database change. Many applications use database to store persistent data. Any change of the database structure will affect program behavior. Modern software applications use object-relational mapping (ORM) to map database table entity objects in a program. When a database structure is changed, we can use ORM to identify which entity objects will be affected, and select test cases referring to the changed entity objects to re-test.

In this study, we use a web application BookStore from GitHub. BookStore is an online shopping platform for searching books, shopping cart, checking out and so on. This web application stores data in MySQL database, using com.mchange.v2.c3p0 package and javax.sql to transform data from MySQL database into entity objects. We made three modifications to the database. The first one changed the name of the "book" table 'name' into 'title'. This change resulted in when initializing the book entity object, the program failed to find the attribute 'name'. Because the "book" table is mapped to the "book" entity object, we could locate the "book" entity in program invariants. Any test case referring to the "book" entity was selected. The second change deleted the whole "category" table. The category is used for categorizing books. After deleting this table, all the books were categorized into one group. Because the "category" table is associated with the "category" entity object, all the test cases referring to this object will be selected. The third modification added a column into the "book" table. Since the new column has not been used , it does not affect the program. The results of these three database modifications are shown in Table 5. For the first modification, our method selected 31 test cases; by retesting all the test cases, three test cases failed and they were by our method. The Recall is 1 the Precision is 9.68%. For the second modification, our method selected 62 test cases, by retesting all the test cases, 12 test cases failed and these 12 test cases were all selected by our method. The recall is 1 the Precision is 19.35%. For the last one, our method selected 31 test cases, by retesting all the test cases, non of them failed.

In summary, our approach successfully selected test cases affected by the database change. The precision is low, because many test cases used mock data instead of using real data in the database.

IV. RELATED WORK

Use either SI (MKS) or CGS as primary units. (SI units are strongly encouraged.) English units may be used as secondary units (in parentheses). This applies to papers in data storage**.** For example, write "15 Gb/cm^2 (100 Gb/in^2)." An exception is when English units are used as identifiers in trade, such as "3½-in disk drive." Avoid combining SI and CGS units, such as current in amperes and magnetic field in oersteds. This often leads to confusion because equations do not balance dimensionally. If you must use mixed units, clearly state the units for each quantity in an equation.

The SI unit for magnetic field strength *H* is A/m. However, if you wish to use units of T, either refer to magnetic flux density *B* or magnetic field strength symbolized as $\mu_0 H$. Use the center dot to separate compound units, e.g., "A·m^2."

V. CONCLUSIONS AND FUTURE WORK

"principle of measurement"). Do not confuse "imply" and "infer."

Prefixes such as "non," "sub," "micro," "multi," and "ultra" are not independent words; they should be joined to the words they modify, usually without a hyphen. There is no period after the "et" in the Latin abbreviation "*et al.*" (it is also italicized). The abbreviation "i.e.," means "that is," and the abbreviation "e.g.," means "for example" (these abbreviations are not italicized).

VI. GUIDELINES FOR GRAPHICS PREPARATION AND SUBMISSION

A. *Types of Graphics*

The following list outlines the different types of graphics published in IEEE journals. They are categorized based on their construction, and use of color / shades of gray:

1) Color/Grayscale figures
Figures that are meant to appear in color, or shades of black/gray. Such figures may include photographs, illustrations, multicolor graphs, and flowcharts.

2) Line Art figures
Figures that are composed of only black lines and shapes. These figures should have no shades or half-tones of gray, only black and white.

3) Author photos
Head and shoulders shots of authors that appear at the end of our papers.

4) Tables
Data charts which are typically black and white, but sometimes include color.

B. *Multipart figures*

Figures compiled of more than one sub-figure presented side-by-side, or stacked. If a multipart figure is made up of multiple figure types (one part is lineart, and another is grayscale or color) the figure should meet the stricter guidelines.

C. *File Formats For Graphics*

Format and save your graphics using a suitable graphics processing program that will allow you to create the images as

PostScript (PS), Encapsulated PostScript (.EPS), Tagged Image File Format (.TIFF), Portable Document Format (.PDF), or Portable Network Graphics (.PNG) sizes them, and adjusts the resolution settings. If you created your source files in one of the following programs you will be able to submit the graphics without converting to a PS, EPS, TIFF, PDF, or PNG file: Microsoft Word, Microsoft PowerPoint, or Microsoft Excel. Though it is not required, it is strongly recommended that these files be saved in PDF format rather than DOC, XLS, or PPT. Doing so will protect your figures from common font and arrow stroke issues that occur when working on the files across multiple platforms. When submitting your final paper, your graphics should all be submitted individually in one of these formats along with the manuscript.

D. Sizing of Graphics

Most charts, graphs, and tables are one column wide (3.5 inches / 88 millimeters / 21 picas) or page wide (7.16 inches / 181 millimeters / 43 picas). The maximum depth a graphic can be is 8.5 inches (216 millimeters / 54 picas). When choosing the depth of a graphic, please allow space for a caption. Figures can be sized between column and page widths if the author chooses, however it is recommended that figures are not sized less than column width unless when necessary.

There is currently one publication with column measurements that do not coincide with those listed above. PROCEEDINGS OF THE IEEE has a column measurement of 3.25 inches (82.5 millimeters / 19.5 picas).

The final printed size of author photographs is exactly 1 inch wide by 1.25 inches tall (25.4 millimeters x 31.75 millimeters / 6 picas x 7.5 picas). Author photos printed in editorials measure 1.59 inches wide by 2 inches tall (40 millimeters x 50 millimeters / 9.5 picas x 12 picas).

E. Resolution

The proper resolution of your figures will depend on the type of figure it is as defined in the "Types of Figures" section. Author photographs, color, and grayscale figures should be at least 300dpi. Line art, including tables should be a minimum of 600dpi.

F. Vector Art

In order to preserve the figures' integrity across multiple computer platforms, we accept files in the following formats: .EPS/.PDF/.PS. All fonts must be embedded or text converted to outlines in order to achieve the best-quality results.

G. Color Space

The term color space refers to the entire sum of colors that can be represented within the said medium. For our purposes, the three main color spaces are Grayscale, RGB (red/green/blue) and CMYK (cyan/magenta/yellow/black). RGB is generally used with on-screen graphics, whereas CMYK is used for printing purposes.

All color figures should be generated in RGB or CMYK color space. Grayscale images should be submitted in Grayscale color space. Line art may be provided in grayscale OR bitmap colorspace. Note that "bitmap colorspace" and "bitmap file format" are not the same thing. When bitmap color space is selected, .TIF/.TIFF/.PNG are the recommended file formats.

H. Accepted Fonts Within Figures

When preparing your graphics IEEE suggests that you use of one of the following Open Type fonts: Times New Roman, Helvetica, Arial, Cambria, and Symbol. If you are supplying EPS, PS, or PDF files all fonts must be embedded. Some fonts may only be native to your operating system; without the fonts embedded, parts of the graphic may be distorted or missing.

A safe option when finalizing your figures is to strip out the fonts before you save the files, creating "outline" type. This converts fonts to artwork what will appear uniformly on any screen.

I. Using Labels Within Figures

1) Figure Axis labels

Figure axis labels are often a source of confusion. Use words rather than symbols. As an example, write the quantity "Magnetization," or "Magnetization *M*," not just "*M*." Put units in parentheses. Do not label axes only with units. As in Fig. 1, for example, write "Magnetization (A/m)" or "Magnetization ($\mathrm{A \cdot m^{-1}}$)," not just "A/m." Do not label axes with a ratio of quantities and units. For example, write "Temperature (K)," not "Temperature/K."

Multipliers can be especially confusing. Write "Magnetization (kA/m)" or "Magnetization (10^3 A/m)." Do not write "Magnetization (A/m) × 1000" because the reader would not know whether the top axis label in Fig. 1 meant 16000 A/m or 0.016 A/m. Figure labels should be legible, approximately 8 to 10 point type.

2) Subfigure Labels in Multipart Figures and Tables

Multipart figures should be combined and labeled before final submission. Labels should appear centered below each subfigure in 8 point Times New Roman font in the format of (a) (b) (c).

J. File Naming

Figures (line artwork or photographs) should be named starting with the first 5 letters of the author's last name. The next characters in the filename should be the number that represents the sequential location of this image in your article. For example, in author "Anderson's" paper, the first three figures would be named ander1.tif, ander2.tif, and ander3.ps.

Tables should contain only the body of the table (not the caption) and should be named similarly to figures, except that '.t' is inserted in-between the author's name and the table number. For example, author Anderson's first three tables would be named ander.t1.tif, ander.t2.ps, ander.t3.eps.

Author photographs should be named using the first five characters of the pictured author's last name. For example, four author photographs for a paper may be named: oppen.ps, moshc.tif, chen.eps, and duran.pdf.

If two authors or more have the same last name, their first initial(s) can be substituted for the fifth, fourth, third... letters of their surname until the degree where there is differentiation. For

example, two authors Michael and Monica Oppenheimer's photos would be named oppmi.tif, and oppmo.eps.

K. Referencing a Figure or Table Within Your Paper

When referencing your figures and tables within your paper, use the abbreviation "Fig." even at the beginning of a sentence. Do not abbreviate "Table." Tables should be numbered with Roman Numerals.

L. Checking Your Figures: The IEEE Graphics Analyzer

The IEEE Graphics Analyzer enables authors to pre-screen their graphics for compliance with IEEE Transactions and Journals standards before submission. The online tool, located at http://graphicsqc.ieee.org/, allows authors to upload their graphics in order to check that each file is the correct file format, resolution, size and colorspace; that no fonts are missing or corrupt; that figures are not compiled in layers or have transparency, and that they are named according to the IEEE Transactions and Journals naming convention. At the end of this automated process, authors are provided with a detailed report on each graphic within the web applet, as well as by email.

For more information on using the Graphics Analyzer or any other graphics related topic, contact the IEEE Graphics Help Desk by e-mail at graphics@ieee.org.

M. Submitting Your Graphics

Because IEEE will do the final formatting of your paper, you do not need to position figures and tables at the top and bottom of each column. In fact, all figures, figure captions, and tables can be placed at the end of your paper. In addition to, or even in lieu of submitting figures within your final manuscript, figures should be submitted individually, separate from the manuscript in one of the file formats listed above in section VI-J. Place figure captions below the figures; place table titles above the tables. Please do not include captions as part of the figures, or put them in "text boxes" linked to the figures. Also, do not place borders around the outside of your figures.

N. *Color Processing / Printing in IEEE Journals*

All IEEE Transactions, Journals, and Letters allow an author to publish color figures on IEEE *Xplore*® at no charge, and automatically convert them to grayscale for print versions. In most journals, figures and tables may alternatively be printed in color if an author chooses to do so. Please note that this service comes at an extra expense to the author. If you intend to have print color graphics, include a note with your final paper indicating which figures or tables you would like to be handled that way, and stating that you are willing to pay the additional fee.

VII. Conclusion

A conclusion section is not required. Although a conclusion may review the main points of the paper, do not replicate the abstract as the conclusion. A conclusion might elaborate on the importance of the work or suggest applications and extensions.

Appendix

Appendixes, if needed, appear before the acknowledgment.

Acknowledgment

The preferred spelling of the word "acknowledgment" in American English is without an "e" after the "g." Use the singular heading even if you have many acknowledgments. Avoid expressions such as "One of us (S.B.A.) would like to thank ... ." Instead, write "F. A. Author thanks ... ." In most cases, sponsor and financial support acknowledgments are placed in the unnumbered footnote on the first page, not here.

References and Footnotes

A. References

References need not be cited in text. When they are, they appear on the line, in square brackets, inside the punctuation. Multiple references are each numbered with separate brackets. When citing a section in a book, please give the relevant page numbers. In text, refer simply to the reference number. Do not use "Ref." or "reference" except at the beginning of a sentence: "Reference [3] shows ... ." Please do not use automatic endnotes in *Word*, rather, type the reference list at the end of the paper using the "References" style.

Reference numbers are set flush left and form a column of their own, hanging out beyond the body of the reference. The reference numbers are on the line, enclosed in square brackets. In all references, the given name of the author or editor is abbreviated to the initial only and precedes the last name. Use them all; use *et al*. only if names are not given. Use commas around Jr., Sr., and III in names. Abbreviate conference titles. When citing IEEE transactions, provide the issue number, page range, volume number, year, and/or month if available. When referencing a patent, provide the day and the month of issue, or application. References may not include all information; please obtain and include relevant information. Do not combine references. There must be only one reference with each number. If there is a URL included with the print reference, it can be included at the end of the reference.

Other than books, capitalize only the first word in a paper title, except for proper nouns and element symbols. For papers published in translation journals, please give the English citation first, followed by the original foreign-language citation See the end of this document for formats and examples of common references. For a complete discussion of references and their formats, see the IEEE style manual at www.ieee.org/authortools.

A. Footnotes

Number footnotes separately in superscripts (Insert |

Footnote).[1] Place the actual footnote at the bottom of the column in which it is cited; do not put footnotes in the reference list (endnotes). Use letters for table footnotes (see Table I).

VIII. Submitting Your Paper for Review

A. Review Stage Using Word 6.0 or Higher

If you want to submit your file with one column electronically, please do the following:

--First, click on the View menu and choose Print Layout.

--Second, place your cursor in the first paragraph. Go to the Format menu, choose Columns, choose one column Layout, and choose "apply to whole document" from the dropdown menu.

--Third, click and drag the right margin bar to just over 4 inches in width.

The graphics will stay in the "second" column, but you can drag them to the first column. Make the graphic wider to push out any text that may try to fill in next to the graphic.

B. Final Stage Using Word 6.0

When you submit your final version (after your paper has been accepted), print it in two-column format, including figures and tables. You must also send your final manuscript on a disk, via e-mail, or through a Web manuscript submission system as directed by the society contact. You may use *Zip* for large files, or compress files using *Compress, Pkzip, Stuffit,* or *Gzip.*

Also, send a sheet of paper or PDF with complete contact information for all authors. Include full mailing addresses, telephone numbers, fax numbers, and e-mail addresses. This information will be used to send each author a complimentary copy of the journal in which the paper appears. In addition, designate one author as the "corresponding author." This is the author to whom proofs of the paper will be sent. Proofs are sent to the corresponding author only.

C. Review Stage Using ScholarOne® Manuscripts

Contributions to the Transactions, Journals, and Letters may be submitted electronically on IEEE's on-line manuscript submission and peer-review system, ScholarOne® Manuscripts. You can get a listing of the publications that participate in ScholarOne at http://www.ieee.org/publications_standards/publications/authors/authors_submission.html First check if you have an existing account. If there is none, please create a new account. After logging in, go to your Author Center and click "Submit First Draft of a New Manuscript."

Along with other information, you will be asked to select the subject from a pull-down list. Depending on the journal, there are various steps to the submission process; you must complete all steps for a complete submission. At the end of each step you must click "Save and Continue"; just uploading the paper is not sufficient. After the last step, you should see a confirmation that the submission is complete. You should also receive an e-mail confirmation. For inquiries regarding the submission of your paper on ScholarOne Manuscripts, please contact oprs-support@ieee.org or call +1 732 465 5861.

ScholarOne Manuscripts will accept files for review in various formats. Please check the guidelines of the specific journal for which you plan to submit.

You will be asked to file an electronic copyright form immediately upon completing the submission process (authors are responsible for obtaining any security clearances). Failure to submit the electronic copyright could result in publishing delays later. You will also have the opportunity to designate your article as "open access" if you agree to pay the IEEE open access fee.

D. Final Stage Using ScholarOne Manuscripts

Upon acceptance, you will receive an email with specific instructions regarding the submission of your final files. To avoid any delays in publication, please be sure to follow these instructions. Most journals require that final submissions be uploaded through ScholarOne Manuscripts, although some may still accept final submissions via email. Final submissions should include source files of your accepted manuscript, high quality graphic files, and a formatted pdf file. If you have any questions regarding the final submission process, please contact the administrative contact for the journal.

In addition to this, upload a file with complete contact information for all authors. Include full mailing addresses, telephone numbers, fax numbers, and e-mail addresses. Designate the author who submitted the manuscript on ScholarOne Manuscripts as the "corresponding author." This is the only author to whom proofs of the paper will be sent.

E. Copyright Form

Authors must submit an electronic IEEE Copyright Form (eCF) upon submitting their final manuscript files. You can access the eCF system through your manuscript submission system or through the Author Gateway. You are responsible for obtaining any necessary approvals and/or security clearances. For additional information on intellectual property rights, visit the IEEE Intellectual Property Rights department web page at http://www.ieee.org/publications_standards/publications/rights/index.html.

IX. IEEE Publishing Policy

The general IEEE policy requires that authors should only submit original work that has neither appeared elsewhere for publication, nor is under review for another refereed publication. The submitting author must disclose all prior publication(s) and current submissions when submitting a manuscript. Do not publish "preliminary" data or results. The submitting author is responsible for obtaining agreement of all coauthors and any consent required from employers or sponsors

[1]It is recommended that footnotes be avoided (except for the unnumbered footnote with the receipt date on the first page). Instead, try to integrate the footnote information into the text.

before submitting an article. The IEEE Transactions and Journals Department strongly discourages courtesy authorship; it is the obligation of the authors to cite only relevant prior work.

The IEEE Transactions and Journals Department does not publish conference records or proceedings, but can publish articles related to conferences that have undergone rigorous peer review. Minimally, two reviews are required for every article submitted for peer review.

X. Publication Principles

The two types of contents of that are published are; 1) peer-reviewed and 2) archival. The Transactions and Journals Department publishes scholarly articles of archival value as well as tutorial expositions and critical reviews of classical subjects and topics of current interest.

Authors should consider the following points:

1) Technical papers submitted for publication must advance the state of knowledge and must cite relevant prior work.
2) The length of a submitted paper should be commensurate with the importance, or appropriate to the complexity, of the work. For example, an obvious extension of previously published work might not be appropriate for publication or might be adequately treated in just a few pages.
3) Authors must convince both peer reviewers and the editors of the scientific and technical merit of a paper; the standards of proof are higher when extraordinary or unexpected results are reported.
4) Because replication is required for scientific progress, papers submitted for publication must provide sufficient information to allow readers to perform similar experiments or calculations and use the reported results. Although not everything need be disclosed, a paper must contain new, useable, and fully described information. For example, a specimen's chemical composition need not be reported if the main purpose of a paper is to introduce a new measurement technique. Authors should expect to be challenged by reviewers if the results are not supported by adequate data and critical details.
5) Papers that describe ongoing work or announce the latest technical achievement, which are suitable for presentation at a professional conference, may not be appropriate for publication.

References

Basic format for books:

J. K. Author, "Title of chapter in the book," in *Title of His Published Book*, *x*th ed. City of Publisher, (only U.S. State), Country: Abbrev. of Publisher, year, ch. *x*, sec. *x*, pp. *xxx–xxx.*

Examples:

[1] G. O. Young, "Synthetic structure of industrial plastics," in *Plastics*, 2nd ed., vol. 3, J. Peters, Ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15–64.

[2] W.-K. Chen, *Linear Networks and Systems*. Belmont, CA, USA: Wadsworth, 1993, pp. 123–135.

Basic format for periodicals:

J. K. Author, "Name of paper," *Abbrev. Title of Periodical*, vol. *x*, *no*. *x*, pp. *xxx-xxx*, Abbrev. Month, year, DOI. 10.1109.*XXX*.123456.

Examples:

[3] J. U. Duncombe, "Infrared navigation—Part I: An assessment of feasibility," *IEEE Trans. Electron Devices*, vol. ED-11, no. 1, pp. 34–39, Jan. 1959, 10.1109/TED.2016.2628402.

[4] E. P. Wigner, "Theory of traveling-wave optical laser," *Phys. Rev*., vol. 134, pp. A635–A646, Dec. 1965.

[5] E. H. Miller, "A note on reflector arrays," *IEEE Trans. Antennas Propagat.*, to be published.

Basic format for reports:

J. K. Author, "Title of report," Abbrev. Name of Co., City of Co., Abbrev. State, Country, Rep. *xxx*, year.

Examples:

[6] E. E. Reber, R. L. Michell, and C. J. Carter, "Oxygen absorption in the earth's atmosphere," Aerospace Corp., Los Angeles, CA, USA, Tech. Rep. TR-0200 (4230-46)-3, Nov. 1988.

[7] J. H. Davis and J. R. Cogdell, "Calibration program for the 16-foot antenna," Elect. Eng. Res. Lab., Univ. Texas, Austin, TX, USA, Tech. Memo. NGL-006-69-3, Nov. 15, 1987.

Basic format for handbooks:

Name of Manual/Handbook, *x* ed., Abbrev. Name of Co., City of Co., Abbrev. State, Country, year, pp. *xxx-xxx.*

Examples:

[8] *Transmission Systems for Communications*, 3rd ed., Western Electric Co., Winston-Salem, NC, USA, 1985, pp. 44–60.

[9] *Motorola Semiconductor Data Manual*, Motorola Semiconductor Products Inc., Phoenix, AZ, USA, 1989.

Basic format for books (when available online):

J. K. Author, "Title of chapter in the book," in *Title of Published Book*, *x*th ed. City of Publisher, State, Country: Abbrev. of Publisher, year, ch. *x*, sec. *x*, pp. *xxx–xxx*. [Online]. Available: http://www.web.com

Examples:

[10] G. O. Young, "Synthetic structure of industrial plastics," in Plastics, vol. 3, Polymers of Hexadromicon, J. Peters, Ed., 2nd ed. New York, NY, USA: McGraw-Hill, 1964, pp. 15-64. [Online]. Available: http://www.bookref.com.

[11] *The Founders' Constitution*, Philip B. Kurland and Ralph Lerner, eds., Chicago, IL, USA: Univ. Chicago Press, 1987. [Online]. Available: http://press-pubs.uchicago.edu/founders/

[12] The Terahertz Wave eBook. ZOmega Terahertz Corp., 2014. [Online]. Available: http://dl.z-thz.com/eBook/zomega_ebook_pdf_1206_sr.pdf. Accessed on: May 19, 2014.

[13] Philip B. Kurland and Ralph Lerner, eds., *The Founders' Constitution.* Chicago, IL, USA: Univ. of Chicago Press, 1987, Accessed on: Feb. 28, 2010, [Online] Available: http://press-pubs.uchicago.edu/founders/

Basic format for journals (when available online):

J. K. Author, "Name of paper," *Abbrev. Title of Periodical*, vol. *x*, no. *x*, pp. *xxx-xxx*, Abbrev. Month, year. Accessed on: Month, Day, year, DOI: 10.1109.*XXX*.123456, [Online].

Examples:

[14] J. S. Turner, "New directions in communications," *IEEE J. Sel. Areas Commun*., vol. 13, no. 1, pp. 11-23, Jan. 1995.

[15] W. P. Risk, G. S. Kino, and H. J. Shaw, "Fiber-optic frequency shifter using a surface acoustic wave incident at an oblique angle," *Opt. Lett.*, vol. 11, no. 2, pp. 115–117, Feb. 1986.

[16] P. Kopyt *et al.,* "Electric properties of graphene-based conductive layers from DC up to terahertz range," *IEEE THz*

Sci. Technol., to be published. DOI: 10.1109/TTHZ.2016.2544142.

Basic format for papers presented at conferences (when available online):

J.K. Author. (year, month). Title. presented at abbrev. conference title. [Type of Medium]. Available: site/path/file

Example:

[17] PROCESS Corporation, Boston, MA, USA. Intranets: Internet technologies deployed behind the firewall for corporate productivity. Presented at INET96 Annual Meeting. [Online]. Available: http://home.process.com/Intranets/wp2.htp

Basic format for reports and handbooks (when available online):

J. K. Author. "Title of report," Company. City, State, Country. Rep. no., (optional: vol./issue), Date. [Online] Available: site/path/file

Examples:

[18] R. J. Hijmans and J. van Etten, "Raster: Geographic analysis and modeling with raster data," R Package Version 2.0-12, Jan. 12, 2012. [Online]. Available: http://CRAN.R-project.org/package=raster

[19] Teralyzer. Lytera UG, Kirchhain, Germany [Online]. Available: http://www.lytera.de/Terahertz_THz_Spectroscopy.php?id=home, Accessed on: Jun. 5, 2014

Basic format for computer programs and electronic documents (when available online):

Legislative body. Number of Congress, Session. (year, month day). *Number of bill or resolution, Title*. [Type of medium]. Available: site/path/file

NOTE: ISO recommends that capitalization follow the accepted practice for the language or script in which the information is given.

Example:

[20] U.S. House. 102nd Congress, 1st Session. (1991, Jan. 11). *H. Con. Res. 1, Sense of the Congress on Approval of Military Action*. [Online]. Available: LEXIS Library: GENFED File: BILLS

Basic format for patents (when available online):

Name of the invention, by inventor's name. (year, month day). Patent Number [Type of medium]. Available: site/path/file

Example:

[21] Musical toothbrush with mirror, by L.M.R. Brooks. (1992, May 19). Patent D 326 189 [Online]. Available: NEXIS Library: LEXPAT File: DES

Basic format for conference proceedings (published):

J. K. Author, "Title of paper," in *Abbreviated Name of Conf.*, City of Conf., Abbrev. State (if given), Country, year, pp. *xxxxxx.*

Example:

[22] D. B. Payne and J. R. Stern, "Wavelength-switched passively coupled single-mode optical network," in *Proc. IOOC-ECOC,* Boston, MA, USA, 1985, pp. 585–590.

Example for papers presented at conferences (unpublished):

[23] D. Ebehard and E. Voges, "Digital single sideband detection for interferometric sensors," presented at the *2nd Int. Conf. Optical Fiber Sensors,* Stuttgart, Germany, Jan. 2-5, 1984.

Basic format for patents:

J. K. Author, "Title of patent," U.S. Patent *x xxx xxx*, Abbrev. Month, day, year.

Example:

[24] G. Brandli and M. Dick, "Alternating current fed power supply," U.S. Patent 4 084 217, Nov. 4, 1978.

Basic format for theses (M.S.) and dissertations (Ph.D.):

a) J. K. Author, "Title of thesis," M.S. thesis, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

b) J. K. Author, "Title of dissertation," Ph.D. dissertation, Abbrev. Dept., Abbrev. Univ., City of Univ., Abbrev. State, year.

Examples:

[25] J. O. Williams, "Narrow-band analyzer," Ph.D. dissertation, Dept. Elect. Eng., Harvard Univ., Cambridge, MA, USA, 1993.

[26] N. Kawasaki, "Parametric study of thermal and chemical nonequilibrium nozzle flow," M.S. thesis, Dept. Electron. Eng., Osaka Univ., Osaka, Japan, 1993.

Basic format for the most common types of unpublished references:

a) J. K. Author, private communication, Abbrev. Month, year.

b) J. K. Author, "Title of paper," unpublished.

c) J. K. Author, "Title of paper," to be published.

Examples:

[27] A. Harrison, private communication, May 1995.

[28] B. Smith, "An approach to graphs of linear forms," unpublished.

[29] A. Brahms, "Representation error for real numbers in binary computer arithmetic," IEEE Computer Group Repository, Paper R-67-85.

Basic formats for standards:

a) *Title of Standard*, Standard number, date.

b) *Title of Standard*, Standard number, Corporate author, location, date.

Examples:

[30] IEEE Criteria for Class IE Electric Systems, IEEE Standard 308, 1969.

[31] Letter Symbols for Quantities, ANSI Standard Y10.5-1968.

Article number in reference examples:

[32] R. Fardel, M. Nagel, F. Nuesch, T. Lippert, and A. Wokaun, "Fabrication of organic light emitting diode pixels by laser-assisted forward transfer," *Appl. Phys. Lett.*, vol. 91, no. 6, Aug. 2007, Art. no. 061103.

[33] J. Zhang and N. Tansu, "Optical gain and laser characteristics of InGaN quantum wells on ternary InGaN substrates," *IEEE Photon. J.*, vol. 5, no. 2, Apr. 2013, Art. no. 2600111

Example when using et al.:

[34] S. Azodolmolky *et al.*, Experimental demonstration of an impairment aware network planning and operation tool for transparent/translucent optical networks," *J. Lightw. Technol.*, vol. 29, no. 4, pp. 439–448, Sep. 2011.

First A. Author (M'76–SM'81–F'87) and all authors may include biographies. Biographies are often not included in conference-related papers. This author became a Member (M) of IEEE in 1976, a Senior Member (SM) in 1981, and a Fellow (F) in 1987. The first paragraph may contain a place and/or date of birth (list place, then date). Next, the author's educational background is listed. The degrees should be listed with type of degree in what field, which institution, city, state, and country, and year the degree was earned. The author's major field of study should be lower-cased.

The second paragraph uses the pronoun of the person (he or she) and not the author's last name. It lists military and work experience, including summer and fellowship jobs. Job titles are capitalized. The current job must have a location; previous positions may be listed without one. Information concerning previous publications may be included. Try not to list more than three books or published articles. The format for listing publishers of a book within the biography is: title of book (publisher name, year) similar to a reference. Current and previous research interests end the paragraph.

The third paragraph begins with the author's title and last name (e.g., Dr. Smith, Prof. Jones, Mr. Kajor, Ms. Hunter). List any memberships in professional societies other than the IEEE. Finally, list any awards and work for IEEE committees and publications. If a photograph is provided, it should be of good quality, and professional-looking. Following are two examples of an author's biography.

Second B. Author was born in Greenwich Village, New York, NY, USA in 1977. He received the B.S. and M.S. degrees in aerospace engineering from the University of Virginia, Charlottesville, in 2001 and the Ph.D. degree in mechanical engineering from Drexel University, Philadelphia, PA, in 2008.

From 2001 to 2004, he was a Research Assistant with the Princeton Plasma Physics Laboratory. Since 2009, he has been an Assistant Professor with the Mechanical Engineering Department, Texas A&M University, College Station. He is the author of three books, more than 150 articles, and more than 70 inventions. His research interests include high-pressure and high-density nonthermal plasma discharge processes and applications, microscale plasma discharges, discharges in liquids, spectroscopic diagnostics, plasma propulsion, and innovation plasma applications. He is an Associate Editor of the journal *Earth, Moon, Planets*, and holds two patents.

Dr. Author was a recipient of the International Association of Geomagnetism and Aeronomy Young Scientist Award for Excellence in 2008, and the IEEE Electromagnetic Compatibility Society Best Symposium Paper Award in 2011.

Third C. Author, Jr. (M'87) received the B.S. degree in mechanical engineering from National Chung Cheng University, Chiayi, Taiwan, in 2004 and the M.S. degree in mechanical engineering from National Tsing Hua University, Hsinchu, Taiwan, in 2006. He is currently pursuing the Ph.D. degree in mechanical engineering at Texas A&M University, College Station, TX, USA.

From 2008 to 2009, he was a Research Assistant with the Institute of Physics, Academia Sinica, Tapei, Taiwan. His research interest includes the development of surface processing and biological/medical treatment techniques using nonthermal atmospheric pressure plasmas, fundamental study of plasma sources, and fabrication of micro- or nanostructured surfaces.

Mr. Author's awards and honors include the Frew Fellowship (Australian Academy of Science), the I. I. Rabi Prize (APS), the European Frequency and Time Forum Award, the Carl Zeiss Research Award, the William F. Meggers Award and the Adolph Lomb Medal (OSA).