Comparative Empirical Evaluation of Software Test Automation using Artificial Intelligence and Manual Test Case Design

Jesús José Bone Caicedo, José Luis Carvajal Carvajal, and Victor Xavier Quiñonez Ku

***Abstract*—This study presents an empirical comparison between manually designed test cases and those automatically generated by generative artificial intelligence tools—ChatGPT and Diffblue Cover—applied to the Spring PetClinic system, developed in Java and Spring Boot. The fully automated experiment comprised 2,480 runs distributed across 12 test classes (6 human and 6 AI-generated), with 40 iterations per class and a total duration of 6.18 hours. Four main metrics—instruction coverage, branch coverage, mutation score, and execution time—were evaluated using descriptive and inferential analysis (Student's t, Welch, and Mann-Whitney U). At the aggregate level (N = 12), no significant differences were found ($p > 0.05$; $d < 0.30$), while at the complete level (N = 2,480), small effects ($r < 0.15$) were observed in all metrics, indicating marginal differences between the two approaches. The results confirm that generative AI can achieve quantitative performance comparable to that of human testing, albeit with more limited functional reasoning. This work provides empirical evidence on the current capabilities and limitations of AI in automated testing, highlighting its potential to accelerate repetitive tasks and improve productivity without replacing human analytical judgment.**

***Index Terms*—automated testing, Diffblue Cover, generative AI, software quality, test case generation**

I. Introduction

In the context of digital transformation and the growing incorporation of artificial intelligence in all stages of the software life cycle, test automation has taken center stage in agile development processes due to its potential to increase coverage, reduce time, and improve product reliability. This technical advance not only responds to a need for efficiency in engineering teams, but also to social demand for more reliable, secure, and highly available digital services, whose quality has a direct impact on sensitive sectors such as health, education, and finance. The emergence of generative AI—with large language models (LLMs) and associated tools—has renewed this field by promising faster and assisted test case generation, albeit with mixed and context-dependent results [1], [2], [3], [4]. In ecosystems dominated by Java and Spring Boot, it is pertinent to subject these solutions to rigorous evaluation to establish their true scope and limitations compared to traditional approaches [1], [2].

Despite recent advances (e.g., improvements in LLM-assisted unit testing in industrial contexts) [5], the literature shows that the landscape is still fragmented. Research such as that by Schäfer et al. [6] and Yang et al. [7] has demonstrated the ability of LLMs to generate automatic unit tests with promising results, but without establishing direct comparisons with human tests. Complementarily, Bhatia et al. [8] empirically evaluated the performance of ChatGPT against traditional tools such as Pynguin, showing comparable or superior coverage, but also a significant proportion of incorrect assertions and dependence on human supervision. Similarly, Kanth et al. [9] contrast conventional methodologies and AI-based approaches, focusing their analysis on efficiency and coverage, although without strict control over a common system under test (SUT). For their part, Kirinuki and Tanno [10] evaluated the interaction between ChatGPT and human testers in black-box testing, highlighting qualitative differences in reasoning and understanding of the system. However, recent systematic and grey literature reviews, such as that by Ricca et al. [11], agree that there is still a lack of empirical evidence combining standardized quantitative metrics—such as mutation score or coverage—with functional observations in the same experimental environment. While recent works have explored advanced metric-driven approaches, such as mutation-guided test generation [12], these efforts remain largely focused on quantitative effectiveness rather than functional interpretation.

In this context, an empirical gap persists: there is a lack of systematic and quantitative comparisons between AI-generated tests and manually designed tests under the same SUT and replicable evaluation framework [2]. Although the literature reports benefits—faster generation and automated exploration—it also warns of limitations associated with variable quality, false positives, and dependence on human supervision [13], [14], [15]. This combination of advances and gaps justifies the need for controlled studies that contrast the actual effectiveness of both approaches under equivalent conditions.

In addition to traditional quantitative indicators, this study incorporates a complementary observation related to functional understanding, understood in empirical terms as the differences observed in the way the compared approaches—generative AI and manual design—address the logic and functionality of the system under test. This dimension does not seek a formal cognitive analysis, but rather to contextualize the numerical results (coverage, mutation score, efficiency) based on how each approach interprets and structures the test cases [2], [4], [14], [15]. Its inclusion enriches the reading of the findings, offering a more complete perspective on the performance and functional quality of the automated testing

J. J. Bone Caicedo, J. L. Carvajal Carvajal, and V. X. Quiñonez Ku are with the Systems and Software Department, Pontifical Catholic University of Ecuador Esmeraldas Campus, Esmeraldas Ecuador (e-mail: jjbone@pucese.edu.ec; jose.carvajal@pucese.edu.ec; xavier.quinonez@pucese.edu.ec).

process.

This study aims to evaluate, in a controlled environment, the effectiveness of test cases automatically generated by generative AI compared to those manually created in Java projects, using a representative Java/Spring Boot project as the system under test (SUT). The overall objective of the study is to evaluate the effectiveness and functional understanding, from an empirical approach, of AI-generated test cases compared to those designed by humans, combining quantitative metrics—coverage (instruction/branch), mutation score, and operational efficiency—with empirical observations on their ability to adequately represent the functionality of the system [16].

The expected contribution integrates comparative evidence with replicable metrics and an analytical component that considers the empirical perspective of functional understanding, offering useful inputs for decision-making in educational and professional settings [1], [5]. In line with this technical and social relevance, the research adopts a comparative experimental approach, based on objective metrics and empirical observations, which allow for the examination of both the quantitative performance and the interpretive quality of AI-generated tests compared to manual tests in a controlled environment. Finally, the article is structured as follows: section 2 describes the materials and methods; section 3 presents the results and discussion; and section 4 develops the conclusions and future projections.

II. The Design, Intent and Limitations of the Templates

The templates are intended to **approximate the final look and page length of the articles/papers**. Therefore, **they are NOT intended to be the final produced work that is displayed in print or on IEEEXplore®**. They will help to give the authors an approximation of the number of pages that will be in the final version. The structure of the LaTeXfiles, as designed, enable easy conversion to XML for the composition systems used by the IEEE's outsource vendors. The XML files are used to produce the final print/IEEEXplore® pdf and then converted to HTML for IEEEXplore®. Have you looked at your article/paper in the HTML version?

III. LaTeX Distributions: Where to Get Them

IEEE recommends using the distribution from the TeXUser Group at http://www.tug.org. You can join TUG and obtain a DVD distribution or download for free from the links provided on their website: http://www.tug.org/texlive/. The DVD includes distributions for Windows, Mac OS X and Linux operating systems.

IV. Where to get the IEEEtran Templates

The **IEEE Template Selector** will always have the most up-to-date versions of the LaTeX and MSWord templates. Please see: https://template-selector.ieee.org/ and follow the steps to find the correct template for your intended publication. Many publications use the IEEETran LaTeX templates, however, some publications have their own special templates. Many of these are based on IEEEtran, but may have special instructions that vary slightly from those in this document.

V. Where to get LaTeX help - user groups

The following on-line groups are very helpful to beginning and experienced LaTeX users. A search through their archives can provide many answers to common questions.

http://www.latex-community.org/
https://tex.stackexchange.com/

VI. Document Class Options in IEEEtran

At the beginning of your LaTeX file you will need to establish what type of publication style you intend to use. The following list shows appropriate documentclass options for each of the types covered by IEEEtran.

Regular Journal Article
```
\documentclass[journal]IEEEtran
```

Conference Paper
```
\documentclass[conference]IEEEtran
```

Computer Society Journal Article
```
\documentclass[10pt,journal,compsoc]IEEEtran
```

Computer Society Conference Paper
```
\documentclass[conference,compsoc]IEEEtran
```

Communications Society Journal Article
```
\documentclass[journal,comsoc]IEEEtran
```

Brief, Correspondence or Technote
```
\documentclass[9pt,technote]IEEEtran
```

There are other options available for each of these when submitting for peer review or other special requirements. IEEE recommends to compose your article in the base 2-column format to make sure all your equations, tables and graphics will fit the final 2-column format. Please refer to the document "IEEEtran_HOWTO.pdf" for more information on settings for peer review submission if required by your EIC.

VII. How to Create Common Front Matter

The following sections describe general coding for these common elements. Computer Society publications and Conferences may have their own special variations and will be noted below.

A. Paper Title

The title of your paper is coded as:

```
\title{The Title of Your Paper}
```

Please try to avoid the use of math or chemical formulas in your title if possible.

B. Author Names and Affiliations

The author section should be coded as follows:

```
\author{Masahito Hayashi
\IEEEmembership{Fellow, IEEE}, Masaki Owari
\thanks{M. Hayashi is with Graduate School
```

```
of Mathematics, Nagoya University, Nagoya,
Japan}
\thanks{M. Owari is with the Faculty of
Informatics, Shizuoka University,
Hamamatsu, Shizuoka, Japan.}
}
```

Be sure to use the \IEEEmembership command to identify IEEE membership status. Please see the "IEEEtran_HOWTO.pdf" for specific information on coding authors for Conferences and Computer Society publications. Note that the closing curly brace for the author group comes at the end of the thanks group. This will prevent you from creating a blank first page.

C. Running Heads

The running heads are declared by using the \markboth command. There are two arguments to this command: the first contains the journal name information and the second contains the author names and paper title.

```
\markboth{Journal of Quantum Electronics,
Vol. 1, No. 1, January 2021}
{Author1, Author2,
\MakeLowercase{\textit{(et al.)}:
Paper Title}
```

D. Copyright Line

For Transactions and Journals papers, this is not necessary to use at the submission stage of your paper. The IEEE production process will add the appropriate copyright line. If you are writing a conference paper, please see the "IEEEtran_HOWTO.pdf" for specific information on how to code "Publication ID Marks".

E. Abstracts

The abstract is the first element of a paper after the \maketitle macro is invoked. The coding is simply:

```
\begin{abstract}
Text of your abstract.
\end{abstract}
```

Please try to avoid mathematical and chemical formulas in the abstract.

F. Index Terms

The index terms are used to help other researchers discover your paper. Each society may have it's own keyword set. Contact the EIC of your intended publication for this list.

```
\begin{IEEEkeywords}
Broad band networks, quality of service
\end{IEEEkeywords}
```

VIII. How to Create Common Body Elements

The following sections describe common body text elements and how to code them.

A. Initial Drop Cap Letter

The first text paragraph uses a "drop cap" followed by the first word in ALL CAPS. This is accomplished by using the \IEEEPARstart command as follows:

```
\IEEEPARstart{T}{his} is the first paragraph
of your paper. . .
```

B. Sections and Subsections

Section headings use standard LaTeX commands: \section, \subsection and \subsubsection. Numbering is handled automatically for you and varies according to type of publication. It is common to not indent the first paragraph following a section head by using \noindent as follows:

```
\section{Section Head}
\noindent The text of your paragraph . . .
```

C. Citations to the Bibliography

The coding for the citations are made with the LaTeX \cite command. This will produce individual bracketed reference numbers in the IEEE style. At the top of your LaTeX file you should include:

```
\usepackage{cite}
```

For a single citation code as follows:

```
see \cite{ams}
```

This will display as: see [**?**]

For multiple citations code as follows:

```
\cite{ams,oxford,lacomp}
```

This will display as [**?**], [**?**], [**?**]

D. Figures

Figures are coded with the standard LaTeX commands as follows:

```
\begin{figure}[!t]
\centering
\includegraphics[width=2.5in]{fig1}
\caption{This is the caption for one fig.}
\label{fig1}
\end{figure}
```

The [!t] argument enables floats to the top of the page to follow IEEE style. Make sure you include:

```
\usepackage{graphicx}
```

at the top of your LaTeXfile with the other package declarations.

To cross-reference your figures in the text use the following code example:

```
See figure \ref{fig1} ...
```

This will produce:
See figure 1 . . .

Fig. 1. This is the caption for one fig.

TABLE I
A SIMPLE TABLE EXAMPLE.

Order of filter	Arbitrary coefficients e_m	coefficients b_{ij}
1	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$
2	$\beta_{22} = (\ 1, -1, -1,\ \ 1,\ \ 1,\ \ 1)$	
3	$b_{ij} = \hat{e}.\hat{\beta_{ij}}$,	$b_{00} = 0$,

E. Tables

Tables should be coded with the standard LaTeX coding. The following example shows a simple table.

```
\begin{table}
\begin{center}
\caption{Filter design equations  ...}
\label{tab1}
\begin{tabular}{| c | c | c |}
\hline
Order & Arbitrary coefficients &
coefficients\\
of filter & $e_m$ &   $b_{ij}$ \\
\hline
1& $b_{ij}=\hat{e}.\hat{\beta_{ij}}$,
& $b_{00}=0$\\
\hline
2&$\beta_{22}=(~1,-1,-1,~~1,~~1,~~1)$ &\\
\hline
3& $b_{ij}=\hat{e}.\hat{\beta_{ij}}$,
& $b_{00}=0$,\\
\hline
\end{tabular}
\end{center}
\end{table}
```

To reference the table in the text, code as follows:

```
Table~\ref{tab1} lists the closed-form...
```

to produce:

Table I lists the closed-form . . .

F. Lists

In this section, we will consider three types of lists: simple unnumbered, numbered and bulleted. There have been numerous options added to IEEEtran to enhance the creation of lists. If your lists are more complex than those shown below, please refer to the "IEEEtran_HOWTO.pdf" for additional options.

A plain unnumbered list

bare_jrnl.tex
bare_conf.tex
bare_jrnl_compsoc.tex
bare_conf_compsoc.tex
bare_jrnl_comsoc.tex

coded as:

```
\begin{list}{}{}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{list}
```

A simple numbered list

1) bare_jrnl.tex
2) bare_conf.tex
3) bare_jrnl_compsoc.tex
4) bare_conf_compsoc.tex
5) bare_jrnl_comsoc.tex

coded as:

```
\begin{enumerate}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{enumerate}
```

A simple bulleted list

- bare_jrnl.tex
- bare_conf.tex
- bare_jrnl_compsoc.tex
- bare_conf_compsoc.tex
- bare_jrnl_comsoc.tex

coded as:

```
\begin{itemize}
\item{bare\_jrnl.tex}
\item{bare\_conf.tex}
\item{bare\_jrnl\_compsoc.tex}
\item{bare\_conf\_compsoc.tex}
\item{bare\_jrnl\_comsoc.tex}
\end{itemize}
```

G. Other Elements

For other less common elements such as Algorithms, Theorems and Proofs, and Floating Structures such as page-

wide tables, figures or equations, please refer to the "IEEEtran_HOWTO.pdf" section on "Double Column Floats."

IX. How to Create Common Back Matter Elements

The following sections demonstrate common back matter elements such as Acknowledgments, Bibliographies, Appendicies and Author Biographies.

A. Acknowledgments

This should be a simple paragraph before the bibliography to thank those individuals and institutions who have supported your work on this article.

```
\section{Acknowledgments}
\noindent Text describing those who
supported your paper.
```

B. Bibliographies

References Simplified: A simple way of composing references is to use the `\bibitem` macro to define the beginning of a reference as in the following examples:

[6] H. Sira-Ramirez. "On the sliding mode control of nonlinear systems," *Systems & Control Letters*, vol. 19, pp. 303–312, 1992.
coded as:

```
\bibitem{Sira3}
H. Sira-Ramirez. ``On the sliding mode
control of nonlinear systems,''
\textit{Systems \& Control Letters},
vol. 19, pp. 303--312, 1992.
```

[7] A. Levant."Exact differentiation of signals with unbounded higher derivatives," in *Proceedings of the 45th IEEE Conference on Decision and Control*, San Diego, California, USA, pp. 5585–5590, 2006.
coded as:

```
\bibitem{Levant}
A. Levant. ``Exact differentiation of
signals with unbounded higher
derivatives,''  in \textit{Proceedings
of the 45th IEEE Conference on
Decision and Control}, San Diego,
California, USA, pp. 5585--5590, 2006.
```

[8] M. Fliess, C. Join, and H. Sira-Ramirez. "Non-linear estimation is easy," *International Journal of Modelling, Identification and Control*, vol. 4, no. 1, pp. 12–27, 2008.
coded as:

```
\bibitem{Cedric}
M. Fliess, C. Join, and H. Sira-Ramirez.
``Non-linear estimation is easy,''
\textit{International Journal of Modelling,
Identification and Control}, vol. 4,
no. 1, pp. 12--27, 2008.
```

[9] R. Ortega, A. Astolfi, G. Bastin, and H. Rodriguez. "Stabilization of food-chain systems using a port-controlled Hamiltonian description," in *Proceedings of the American Control Conference*, Chicago, Illinois, USA, pp. 2245–2249, 2000.
coded as:

```
\bibitem{Ortega}
R. Ortega, A. Astolfi, G. Bastin, and H.
Rodriguez. ``Stabilization of food-chain
systems using a port-controlled Hamiltonian
description,'' in \textit{Proceedings of the
American Control Conference}, Chicago,
Illinois, USA, pp. 2245--2249, 2000.
```

C. Accented Characters in References

When using accented characters in references, please use the standard LaTeX coding for accents. **Do not use math coding for character accents**. For example:

```
\'e, \"o, \`a, \~e
```

will produce: é, ö, à, ẽ

D. Use of BibTeX

If you wish to use BibTeX, please see the documentation that accompanies the IEEEtran Bibliography package.

E. Biographies and Author Photos

Authors may have options to include their photo or not. Photos should be a bit-map graphic (.tif or .jpg) and sized to fit in the space allowed. Please see the coding samples below:

```
\begin{IEEEbiographynophoto}{Jane Doe}
Biography text here without a photo.
\end{IEEEbiographynophoto}
```

or a biography with a photo

```
\begin{IEEEbiography}[{\includegraphics
[width=1in,height=1.25in,clip,
keepaspectratio]{fig1.png}}]
{IEEE Publications Technology Team}
In this paragraph you can place
your educational, professional background
and research and other interests.
\end{IEEEbiography}
```

Please see the end of this document to see the output of these coding examples.

X. Mathematical Typography and Why It Matters

Typographical conventions for mathematical formulas have been developed to **provide uniformity and clarity of presentation across mathematical texts**. This enables the readers of those texts to both understand the author's ideas and to grasp new concepts quickly. While software such as LaTeX and MathType® can produce aesthetically pleasing math when

used properly, it is also very easy to misuse the software, potentially resulting in incorrect math display.

IEEE aims to provide authors with the proper guidance on mathematical typesetting style and assist them in writing the best possible article.

As such, IEEE has assembled a set of examples of good and bad mathematical typesetting. You will see how various issues are dealt with. The following publications have been referenced in preparing this material:

Mathematics into Type, published by the American Mathematical Society
The Printing of Mathematics, published by Oxford University Press
The LaTeXCompanion, by F. Mittelbach and M. Goossens
More Math into LaTeX, by G. Grätzer
AMS-StyleGuide-online.pdf, published by the American Mathematical Society

Further examples can be seen at http://journals.ieeeauthorcenter.ieee.org/wp-content/uploads/sites/7/IEEE-Math-Typesetting-Guide.pdf

A. Display Equations

A simple display equation example shown below uses the "equation" environment. To number the equations, use the `\label` macro to create an identifier for the equation. LaTeX will automatically number the equation for you.

$$x = \sum_{i=0}^{n} 2iQ. \tag{1}$$

is coded as follows:

```
\begin{equation}
\label{deqn_ex1}
x = \sum_{i=0}^{n} 2{i} Q.
\end{equation}
```

To reference this equation in the text use the `\ref` macro. Please see (1)
is coded as follows:

```
Please see (\ref{deqn_ex1})
```

B. Equation Numbering

Consecutive Numbering: Equations within an article are numbered consecutively from the beginning of the article to the end, i.e., (1), (2), (3), (4), (5), etc. Do not use roman numerals or section numbers for equation numbering.

Appendix Equations: The continuation of consecutively numbered equations is best in the Appendix, but numbering as (A1), (A2), etc., is permissible.

Hyphens and Periods: Hyphens and periods should not be used in equation numbers, i.e., use (1a) rather than (1-a) and (2a) rather than (2.a) for sub-equations. This should be consistent throughout the article.

C. Multi-line equations and alignment

Here we show several examples of multi-line equations and proper alignments.

A single equation that must break over multiple lines due to length with no specific alignment.

$$\begin{multline}\text{The first line of this example}\\ \text{The second line of this example}\\ \text{The third line of this example}\end{multline} \tag{2}$$

is coded as:

```
\begin{multline}
\text{The first line of this example}\\
\text{The second line of this example}\\
\text{The third line of this example}
\end{multline}
```

A single equation with multiple lines aligned at the = signs

$$a = c + d \tag{3}$$
$$b = e + f \tag{4}$$

is coded as:

```
\begin{align}
a &= c+d \\
b &= e+f
\end{align}
```

The `align` environment can align on multiple points as shown in the following example:

$$x = y \qquad X = Y \qquad a = bc \tag{5}$$
$$x' = y' \qquad X' = Y' \qquad a' = bz \tag{6}$$

is coded as:

```
\begin{align}
x &= y & X & =Y & a &=bc\\
x' &= y' & X' &=Y' &a' &=bz
\end{align}
```

D. Subnumbering

The amsmath package provides a `subequations` environment to facilitate subnumbering. An example:

$$f = g \tag{7a}$$
$$f' = g' \tag{7b}$$
$$\mathcal{L}f = \mathcal{L}g \tag{7c}$$

is coded as:

```
\begin{subequations}\label{eq:2}
\begin{align}
f&=g \label{eq:2A}\\
f' &=g' \label{eq:2B}\\
\mathcal{L}f &= \mathcal{L}g \label{eq:2c}
\end{align}
\end{subequations}
```

E. Matrices

There are several useful matrix environments that can save you some keystrokes. See the example coding below and the output.

A simple matrix:

$$\begin{matrix} 0 & 1 \\ 1 & 0 \end{matrix} \tag{8}$$

is coded as:

```
\begin{equation}
\begin{matrix}  0 &  1 \\
1 &  0 \end{matrix}
\end{equation}
```

A matrix with parenthesis

$$\begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix} \tag{9}$$

is coded as:

```
\begin{equation}
\begin{pmatrix} 0 & -i \\
 i &  0 \end{pmatrix}
\end{equation}
```

A matrix with square brackets

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \tag{10}$$

is coded as:

```
\begin{equation}
\begin{bmatrix} 0 & -1 \\
1 &  0 \end{bmatrix}
\end{equation}
```

A matrix with curly braces

$$\begin{Bmatrix} 1 & 0 \\ 0 & -1 \end{Bmatrix} \tag{11}$$

is coded as:

```
\begin{equation}
\begin{Bmatrix} 1 &  0 \\
0 & -1 \end{Bmatrix}
\end{equation}
```

A matrix with single verticals

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} \tag{12}$$

is coded as:

```
\begin{equation}
\begin{vmatrix} a &  b \\
c &  d \end{vmatrix}
\end{equation}
```

A matrix with double verticals

$$\begin{Vmatrix} i & 0 \\ 0 & -i \end{Vmatrix} \tag{13}$$

is coded as:

```
\begin{equation}
\begin{Vmatrix} i &  0 \\
0 & -i \end{Vmatrix}
\end{equation}
```

F. Arrays

The `array` environment allows you some options for matrix-like equations. You will have to manually key the fences, but you'll have options for alignment of the columns and for setting horizontal and vertical rules. The argument to `array` controls alignment and placement of vertical rules.

A simple array

$$\left(\begin{array}{cccc} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{array} \right) \tag{14}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccc}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

A slight variation on this to better align the numbers in the last column

$$\left(\begin{array}{cccc} a+b+c & uv & x-y & 27 \\ a+b & u+v & z & 134 \end{array} \right) \tag{15}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{cccr}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

An array with vertical and horizontal rules

$$\left(\begin{array}{c|c|c|r} a+b+c & uv & x-y & 27 \\ \hline a+b & u+v & z & 134 \end{array} \right) \tag{16}$$

is coded as:

```
\begin{equation}
\left(
\begin{array}{c|c|c|r}
a+b+c & uv & x-y & 27\\
a+b & u+v & z & 134
\end{array} \right)
\end{equation}
```

Note the argument now has the pipe "|" included to indicate the placement of the vertical rules.

G. Cases Structures

Many times we find cases coded using the wrong environment, i.e., `array`. Using the `cases` environment will save keystrokes (from not having to type the `\left\lbrace`) and automatically provide the correct column alignment.

$$z_m(t) = \begin{cases} 1, & \text{if } \beta_m(t) \\ 0, & \text{otherwise.} \end{cases}$$

is coded as follows:

```
\begin{equation*}
{z_m(t)} =
\begin{cases}
1,&{\text{if}}\ {\beta }_m(t),\\
{0,}&{\text{otherwise.}}
\end{cases}
\end{equation*}
```

Note that the "&" is used to mark the tabular alignment. This is important to get proper column alignment. Do not use `\quad` or other fixed spaces to try and align the columns. Also, note the use of the `\text` macro for text elements such as "if" and "otherwise".

H. Function Formatting in Equations

In many cases there is an easy way to properly format most common functions. Use of the `\` in front of the function name will in most cases, provide the correct formatting. When this does not work, the following example provides a solution using the `\text` macro.

$$d_R^{KM} = \underset{d_l^{KM}}{\arg\min}\{d_1^{KM}, \ldots, d_6^{KM}\}.$$

is coded as follows:

```
\begin{equation*}
 d_{R}^{KM} = \underset {d_{l}^{KM}}
 {\text{arg min}} \{ d_{1}^{KM},
 \ldots,d_{6}^{KM}\}.
\end{equation*}
```

I. Text Acronyms inside equations

This example shows where the acronym "MSE" is coded using `\text{}` to match how it appears in the text.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2$$

```
\begin{equation*}
 \text{MSE} = \frac {1}{n}\sum _{i=1}^{n}
(Y_{i} - \hat {Y_{i}})^{2}
\end{equation*}
```

J. Obsolete Coding

Avoid the use of outdated environments, such as `eqnarray` and $$ math delimiters, for display equations. The $$ display math delimiters are left over from PlainTeX and should not be used in LaTeX, ever. Poor vertical spacing will result.

K. Use Appropriate Delimiters for Display Equations

Some improper mathematical coding advice has been given in various YouTube™ videos on how to write scholarly articles, so please follow these good examples:

For **single-line unnumbered display equations**, please use the following delimiters:

```
\[ . . . \] or
\begin{equation*} . . . \end{equation*}
```

Note that the * in the environment name turns off equation numbering.

For **multiline unnumbered display equations** that have alignment requirements, please use the following delimiters:

```
\begin{align*} . . . \end{align*}
```

For **single-line numbered display equations**, please use the following delimiters:

```
\begin{equation} . . . \end{equation}
```

For **multiline numbered display equations**, please use the following delimiters:

```
\begin{align} . . . \end{align}
```

XI. LaTeX Package Suggestions

Immediately after your documenttype declaration at the top of your LaTeX file is the place where you should declare any packages that are being used. The following packages were used in the production of this document.

```
\usepackage{amsmath,amsfonts}
\usepackage{algorithmic}
\usepackage{array}
\usepackage[caption=false,font=normalsize,
   labelfont=sf,textfont=sf]{subfig}
\u00sepackage{textcomp}
\usepackage{stfloats}
\usepackage{url}
\usepackage{verbatim}
\usepackage{graphicx}
\usepackage{balance}
```

XII. Additional Advice

Please use "soft" (e.g., `\eqref{Eq}`) or `(\ref{Eq})` cross references instead of "hard" references (e.g., `(1)`). That will make it possible to combine sections, add equations, or change the order of figures or citations without having to go through the file line by line.

Please note that the `{subequations}` environment in LaTeX will increment the main equation counter even when there are no equation numbers displayed. If you forget that, you might write an article in which the equation numbers skip from (17) to (20), causing the copy editors to wonder if you've discovered a new method of counting.

BIBTEX does not work by magic. It doesn't get the bibliographic data from thin air but from .bib files. If you use BIBTEX to produce a bibliography you must send the .bib files.

LATEX can't read your mind. If you assign the same label to a subsubsection and a table, you might find that Table I has been cross referenced as Table IV-B3.

LATEX does not have precognitive abilities. If you put a `\label` command before the command that updates the counter it's supposed to be using, the label will pick up the last counter to be cross referenced instead. In particular, a `\label` command should not go before the caption of a figure or a table.

Please do not use `\nonumber` or `\notag` inside the `{array}` environment. It will not stop equation numbers inside `{array}` (there won't be any anyway) and it might stop a wanted equation number in the surrounding equation.

XIII. A Final Checklist

1) Make sure that your equations are numbered sequentially and there are no equation numbers missing or duplicated. Avoid hyphens and periods in your equation numbering. Stay with IEEE style, i.e., (1), (2), (3) or for subequations (1a), (1b). For equations in the appendix (A1), (A2), etc..
2) Are your equations properly formatted? Text, functions, alignment points in cases and arrays, etc.
3) Make sure all graphics are included.
4) Make sure your references are included either in your main LaTeX file or a separate .bib file if calling the external file.

References

[1] A. Trudova, M. Dolezel, and A. Buchalcevova, "Artificial intelligence in software test automation: A systematic literature review," pp. 181–192, 2020.

[2] V. Garousi, Z. Jafarov, A. B. Keleş, S. Değirmenci, E. Özdemir Testinium AŞ, and R. Zarringhalami, "Ai-powered software testing tools: A systematic review and empirical assessment of their features and limitations," Tech. Rep., 5 2025. [Online]. Available: https://arxiv.org/abs/2409.00411

[3] A. Singh and O. Al-Azzam, "Artificial intelligence applied to software testing." Academy and Industry Research Collaboration Center (AIRCC), 11 2023, pp. 01–12.

[4] T. Li, C. Cui, R. Huang, D. Towey, and L. Ma, "Large language models for automated web-form-test generation: An empirical study," *ACM Transactions on Software Engineering and Methodology*, 5 2025. [Online]. Available: https://doi.org/10.1145/3735553

[5] N. Alshahwan, J. Chheda, A. Finegenova, B. Gokkaya, M. Harman, I. Harper, A. Marginean, S. Sengupta, and E. Wang, "Automated unit test improvement using large language models at meta," in *FSE Companion - Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering*. Association for Computing Machinery, Inc, 2024, pp. 185–196. [Online]. Available: https://arxiv.org/abs/2402.09171

[6] M. Schäfer, S. Nadi, A. Eghbali, and F. Tip, "An empirical evaluation of using large language models for automated unit test generation," 12 2023. [Online]. Available: https://arxiv.org/abs/2302.06527

[7] L. Yang, C. Yang, S. Gao, W. Wang, B. Wang, Q. Zhu, X. Chu, J. Zhou, G. Liang, Q. Wang, and J. Chen, "An empirical study of unit test generation with large language models." 9 2024. [Online]. Available: https://arxiv.org/abs/2406.18181

[8] S. Bhatia, T. Gandhi, D. Kumar, and P. Jalote, "Unit test generation using generative ai : A comparative performance analysis of autogeneration tools," 2 2024. [Online]. Available: https://arxiv.org/abs/2312.10622

[9] R. Kanth, R. Guru, M. B. K, and D. Akshaya, "Ai vs. conventional testing: A comprehensive comparison of effectiveness &efficiency," *Educational Administration: Theory and Practice*, 1 2023. [Online]. Available: https://kuey.net/index.php/kuey/article/view/7495

[10] H. Kirinuki and H. Tanno, "Chatgpt and human synergy in black-box testing: A comparative analysis," 1 2024. [Online]. Available: https://arxiv.org/abs/2401.13924

[11] F. Ricca, A. Marchetto, and A. Stocco, "A multi-year grey literature review on ai-assisted test automation," 1 2025. [Online]. Available: https://arxiv.org/abs/2408.06224

[12] G. Wang, Q. Xu, L. C. Briand, and K. Liu, "Mutation-guided unit test generation with a large language model," 8 2025. [Online]. Available: https://arxiv.org/abs/2506.02954

[13] R. F. de Lima Junior, L. F. P. de Barros Presta, L. S. Borborema, V. N. da Silva, M. L. de Melo Dahia, and A. C. S. e Santos, "A case study on test case construction with large language models: Unveiling practical insights and challenges," 12 2023. [Online]. Available: https://arxiv.org/abs/2312.12598

[14] S. Rehan, B. Al-Bander, and A. A.-S. Ahmad, "Harnessing large language models for automated software testing: A leap towards scalable test case generation," *Electronics (Switzerland)*, vol. 14, 4 2025. [Online]. Available: https://www.mdpi.com/2079-9292/14/7/1463

[15] D. Huang, J. M. Zhang, M. Harman, Q. Zhang, M. Du, and S.-K. Ng, "Benchmarking llms for unit test generation from real-world functions," 8 2025. [Online]. Available: https://arxiv.org/abs/2508.00408

[16] M. Gkikopouli and B. Bataa, "Empirical comparison between conventional and ai-based automated unit test generation tools in java," p. 30, 6 2023. [Online]. Available: https://www.diva-portal.org/smash/record.jsf?pid=diva2%3A1764443&dswid=4942

Jane Doe Biography text here without a photo.

IEEE Publications Technology Team In this paragraph you can place your educational, professional background and research and other interests.