Guidelines for Collaboration

Zhiyang Ong *

February 1, 2018

Abstract

This is a set of guidelines for conduct while collaborating on open source projects. It also includes guidelines for creating a shared BibTeX database.

Revision History

Revision history:

1. Version 1, October 2, 2014. Initial version of the guideline (for another project).
2. Version 1.1, December 23, 2014. Version ported for this boilerplate code project.
3. Version 2, October 20, 2015. Added guidelines for `Doxygen`-supported, `Javadoc`-based coding standard. This coding standard is also known as coding style, coding style guide, coding guideline, coding scheme, code convention, code documentation guideline, programming guideline, or programming style.
4. Version 2.1, October 21, 2015. Finished guidelines for `Doxygen`-supported, `Javadoc`-based coding standard for *C++*.
5. Version 2.2, June 4, 2016. Finished section for additional guidelines: to include documentation using *Markdown*, and tools for software development, integrated circuit and cyber-physical system design, and documentation.
6. Version 3, November 3, 2016. Added guidelines for: documenting `GNU Octave` and MATLAB code, in order to facilitate documentation generation using *Texinfo* [64–66, 78]; sharing of source code, design files, sets of benchmarks, data sets, and documentation on online repositories [21, 25]; and added section on exception safety.
7. Version 3.1, November 4, 2016. Fixed references for indent style conventions.
8. Version 3.2, December 20, 2016. Update guidelines for conduct.
9. Version 3.3, February 3, 2017. Update information about usage of *GitHub*'s services.
10. Version 3.4, March 11, 2017. Update information on naming convention.
11. Version 3.5, October 9, 2017. Update guidelines on commenting/writing code.
12. Version 3.6, December 24, 2017. Fix grammatical error in a sentence.
13. Version 3.7, January 25, 2018. Added suggestions for software architecture of my computer programs.
14. Version 3.8, January 31, 2018. Added information about coding style guideline for different computer languages, and also about online repositories that facilitate research reproducibility, replicability, and repeatability.
15. Version 3.9, February 1, 2018. Added information about developing software in a *Python*ic style.

*Email correspondence to: ongz@acm.org

1 Guidelines for Conduct

Members of the open source software and/or hardware projects should follow the *Code of Conduct* of the *Institute of Electrical and Electronics Engineers (IEEE)* [27–29] and the *Association for Computing Machinery (ACM)* [3]. Also, actions of discrimination are not acceptable [30]; we should intentionally commit to inclusive diversity. An additional guideline is "Dave Packard's 11 simple rules" [13].

In addition, when there is a dispute about which technology, algorithm, design paradigm/style/pattern, process, or methodology to use, follow the "Code Wins Arguments" philosophy [39,81]. Also, when considerable effort has been invested in an automated regression testing/verification infrastructure, do not be afraid to "move fast and break things" [19,23].

Lastly, we should adopt a mission-focused and value-based approach to participate in meetings and discussions for the project(s). We should be flexible/liberal enough to consider and explore viable alternate approaches to do things and solve problems. Where disputes occur, a data-driven, fact-based approach based on the "Code Wins Arguments" philosophy should be used to resolve conflicts.

2 Guidelines for Creating a Shared BibTeX Database

Guidelines for creating BibTeX entries and the BibTeX database, which is used for writing the paper, are given as follows:

1. Each BibTeX key should be unique:
   (a) Check if your desired BibTeX key already exists in the BibTeX database.
   (b) Use the following format for creating BibTeX keys: [first] author's last name, appended by the year of publication. E.g., my first conference paper would have the BibTeX key Ong2014. If the year of publication is not known, use an approximate year, with XY for the last 2 digits in the year (e.g., 20XY). Alternatively, if you cannot determine if it was published this millennium or the previous millennium, use UNKNOWN. For example, use Smith20XY, or KleinbergUNKNOWN.
   (c) Remove duplicate entries in the BibTeX database. **WARNING! Before doing this, perform a union operation on the fields of the BibTeX entries. For example, if a BibTeX entry has information that the other BibTeX entry does not have, and vice versa, merge the information to a BibTeX entry.**
   (d) **Rationale: Duplicate BibTeX entries will cause problems in typesetting.**
   (e) Regarding hash collision of BibTeX keys, such as multiple instances of Gratz2014, distinguish them by appending a letter to them. E.g., use Gratz2014a, Gratz2014b, Gratz2014c, and so on. If we run out of letters, append it with "a" followed by a number. The use of the letter "a" separates the year from the instance of BibTeX key. That is, Gratz2014a2 tells me that it is the 29^{th} instance of Gratz2014, as opposed to Gratz201429.
   (f) If possible, restrict the characters of each BibTeX key to be alphanumeric. The year is always numeric, and is appended to the (first) author's last name.
      i. If the (first) author's last name has characters with accents, trim the characters used to typeset the accents from the (first) author's last name, and append the year of publication to it. E.g., *Sǒménzi* (year 2000) becomes *Somenzi2000*.
      ii. If the (first) author's last name has characters that are not letters in English, anglicize those characters. We should avoid using the transliteration for a given non-English

language, since such transliteration may not be standardized (for non-commonly spoken/used languages). Also, supporting letters from other languages is a tedious task. Hence, we can use the anglicized version of their last names instead.

2. For terms that should be typeset as is, place them in between braces (i.e., curly brackets). That is, put curly braces around acronyms and mixed-case names.
   (a) For example, terms in upper or mixed cases (upper and lower cases), such as names (e.g., McMullen) and acronyms (e.g., SIGDA), place them in between braces (i.e., {McMullen} and {SIGDA}). This prevents the titles (or another BibTeX field) from changing the term into lower case, with exception for the first term/word. E.g., "ICCAD Update: A Report from SIGDA" may typeset into "ICCAD Update: A report from sigda".
3. For special symbols that are typeset with LaTeX in the `math mode`, such as α, place them in between a pair of dollar signs (i.e., α).
4. For each BibTeX entry, check if all required fields are complete. See pages 8 and 9 in §3.1 of [53] for a list of BibTeX entry types; alternatively, refer to the *Wikipedia* entry for , or [38, §12.2.1, pp. 230–231]. In this/these list(s), the required fields are listed for each BibTeX entry.
5. For the `pages` field, ensure that all page ranges are indicated with double hyphens. E.g., "page = {11--34},". This makes the page range looks more pretty.
6. For the `pages` field, ensure that multiple pages and/or page ranges are separated by commas. E.g., "page = {11-34, 57, 88, 109--187},".
7. For books and journal articles that have an associated digital object identifier (DOI) [32], ensure that the `doi` field is included in the BibTeX entry with the DOI of the publication. This makes it easier for people to access the web page for the book or journal/conference paper.
8. Stylistic validation of the references can be carried out as follows:
   (a) Include all BibTeX keys in one citation in your LaTeX document.
   (b) Typeset the LaTeX document.
   (c) Check that the font and style of the reference list is correct.
   (d) If there are errors, correct the errors as appropriate.
   (e) Finally, the BibTeX database should be correct.
9. Information that I would include when citing common sources of information, such as *Wikipedia*, using the Harvard Referencing Style:
   (a) Wikipedia contributors, "TITLE_OF_THE_ARTICLE," in {\it Wikipedia, The Free Encyclopedia: CATEGORY}, Wikimedia Foundation, San Francisco, CA, MONTH DATE, YEAR. Available online at: \url{URL}; last accessed on August 26, 2014.
   (b) Wikibooks contributors, "CHAPTER_NAME," in {\it TITLE_OF_THE_BOOK}, Wikibooks: Open books for an open world, Wikimedia Foundation, San Francisco, CA, MONTH DATE, YEAR. Available online at: \url{URL}; last accessed on August 26, 2014.
   (c) Wikibooks contributors, "TITLE_OF_THE_BOOK," Wikibooks: Open books for an open world, Wikimedia Foundation, San Francisco, CA, MONTH DATE, YEAR. Available online at: \url{URL}; last accessed on August 26, 2014.
   (d) Wiktionary contributors, "TITLE," Wiktionary, Wikimedia Foundation, San Francisco, CA, MONTH DATE, YEAR. Available online at: \url{URL}; last accessed on August 26, 2014.
   (e) Dictionary.com, "WORD," IAC, Oakland, CA, MONTH DATE, YEAR. Available online at: \url{URL}; last accessed on August 26, 2014.
   (f) AUTHOR, "TITLE," in {\it The New York Times: The Opinion Pages: Op-Ed Contributor}, The New York Times Company, New York, NY, MONTH DATE, YEAR. Available online at: \url{URL}; last accessed on August 26, 2014.

(g) When BibTeX entries are created for the aforementioned sources of information, populate the appropriate fields so that each information in the aforementioned sources are included in the BibTeX entries.

10. Refer to the file "bibtex-template.txt" for templates for selected BibTeX entry types. The more information that you can put in, the easier you can protect yourself from accusations of plagiarism and to make it easier for people (including yourself) to find the reference again. This is especially true for web-based references/resources.

3 Recommended Fields for BibTeX Entries

The recommended fields for BibTeX entries are:

1. techreport:
   (a) Address
   (b) Author
   (c) Howpublished
   (d) Institution
   (e) Keywords
   (f) Month
   (g) Number
   (h) Title
   (i) Url
   (j) Year
2. proceedings:
   (a) Address
   (b) DOI
   (c) Editor
   (d) Keywords
   (e) Month
   (f) Organization
   (g) Publisher
   (h) Series
   (i) Title
   (j) Volume
   (k) Year
3. manual:
   (a) Address
   (b) Author
   (c) Howpublished
   (d) Keywords
   (e) Month
   (f) Organization
   (g) Title
   (h) Url
   (i) Year
4. incollection:

(a) Address
(b) Author
(c) Booktitle
(d) Chapter
(e) DOI
(f) Edition
(g) Howpublished
(h) Keywords
(i) Pages
(j) Publisher
(k) Series
(l) Title
(m) Url
(n) Volume
(o) Year

5. inproceedings:
   (a) Address
   (b) Author
   (c) Booktitle
   (d) DOI
   (e) Keywords
   (f) Month
   (g) Organization
   (h) Pages
   (i) Publisher
   (j) Series
   (k) Title
   (l) Volume
   (m) Year
6. article:
   (a) Address
   (b) Author
   (c) DOI
   (d) Journal
   (e) Keywords
   (f) Month
   (g) Number
   (h) Pages
   (i) Publisher
   (j) Title
   (k) Volume
   (l) Year
7. phdthesis (or mastersthesis):
   (a) Address
   (b) Author

   (c) Howpublished
   (d) Keywords
   (e) Month
   (f) Number
   (g) School
   (h) Title
   (i) Url
   (j) Year

8. misc:
   (a) Address
   (b) Author
   (c) Howpublished
   (d) Keywords
   (e) Month
   (f) Publisher or School
   (g) Title
   (h) Url
   (i) Year

9. book:
   (a) Address
   (b) Author
   (c) DOI
   (d) Edition
   (e) Keywords
   (f) Month
   (g) Pages
   (h) Publisher
   (i) Series
   (j) Title
   (k) Volume
   (l) Year

4 Coding Standard

This is a guideline for `Doxygen`-supported [70], `Javadoc`-based [42] coding standard that shall be used for this boilerplate code project. The term "coding standard" is used interchangeably/synonymously with coding style, coding style guide, coding guideline, coding scheme, code convention, code documentation guideline, programming guideline, or programming style. My coding style/standard shall be self-documenting. The documentation generator that shall be supported is: `Doxygen`. Since I am using `Doxygen` for generating documentation, I can use LaTeX to provide richer markup.

Document the known bugs for each function/method.

My indent style would be the *1TBS* variant of the *K&R* style, which is an abbreviation of "*The One True Brace Style*". It is also equivalent to the *Kernel Normal Form style* (or *BSD KNF style*) [75].

Classes, functions/methods, constants, macros, and static and instance variables shall be named using complete words or well-known abbreviations that are concatenated with an underscore in *C++*; this is a deviation from the *Hungarian notation* that uses an upper case letter to distinguish words/abbreviations in the name (i.e., the Start case style of writing; see letter case). That is, the naming convention followed is using multiple-word identifiers, via delimiter-separated words rather than letter-case separated words (e.g., Hungarian notation) [79].

For *C++* programs, the following tags shall be used in the comments:

1. @author *Author's_Name*: indicate the author (*Author's_Name*) of the file/function
2. @version *X.Y*: indicate the version (*X.Y*) of the file
3. @section *SECTION_NAME*: indicate the section (*SECTION_NAME*) of the file, which can be: *LICENSE* or *DESCRIPTION*
4. @param *x*: indicate the parameter (*x*) of the constructor or function
5. @exception *Exception_Name*, or @throws *Exception_Name*: an exception that a function/method can throw
6. @return *Return_Statement*: indicate the return (type and) action of the function
7. @see *reference*: a link to another element in the documentation; e.g., @see *Class_Name*, or @see *Class_Name#member_function_name*
8. @since *X.Y: Month-Day-Year*: This functionality has been added since version *X.Y* (and on the date *Month-Day-Year*)
9. @deprecated *description*: Describe an outdated function/method, and indicate when the function/method has deprecated
10. "@link ... *URL...* @endlink" is used to include hyperlinks in the generated documentation for Doxygen
11. #### IMPORTANT NOTES: Notes that are critical for helping the reader understanding assumptions and decisions made while developing the software
12. @todo(<message>, <version>) (or #### TO BE COMPLETED): Task to be finished at a later time
13. #### TO BE FIXED: Task to be debugged at a later time
14. @migration(<message>, <version>): Code is being migrated to another function/method, or class.
15. See http://www.stack.nl/~dimitri/doxygen/commands.html for more information of tags that are recognized by `Doxygen`.
16. @pre (or @precondition): Precondition(s) of the function.
17. @assert (or @assertion): Assertion(s) of the function.
18. @post (or @postcondition): Postcondition(s) of the function.

The order of tags in different sections of the *C++* code is given as follows:

1. Headers/Interfaces and Classes: @version, @author, @since, @link, @todo, @deprecated, @migration, and @see
2. Constructors: @param, @throws, @since, @link, @todo, @deprecated, @migration, and @see. For collaborators modifying or extending my code, they should include the @version and @author tags before the @param tag(s).
3. Functions/Methods: @param, @pre, @assert, @post, @return, @throws, @since, @link, @todo, @deprecated, @migration, and @see. For collaborators modifying or extending my code, they should include the @version and @author tags before the @param tag(s).
4. Variables can use the @see tags.

5. The @deprecated tag can be used for headers/interfaces, classes, constructors, functions/methods, and variables.

Additional coding style guidelines can be found in [5, 10, 26, 46, 55, 61, 68].

For a suggested coding style for *Python* and *Ruby* scripts, see [71] and [43], respectively. Regarding coding style guidelines for embedded *C*, see [4, 40]. In addition, there exists coding style guidelines for *Java* [11, 42, 48–51, 63] and *LabVIEW* [8, 14]. Coding style guidelines for *Verilog* can be found at: [6, 7]. Likewise, the coding style guide for *SystemVerilog* can be found at [45]. For other coding style guidelines, see [9, 20, 22, 31, 34, 37, 41, 44, 59, 60, 62, 69, 72, 80].

While well-documented source code is desired, natural language programming [76] is usually infeasible due to the choices of programming/computer languages used. Also, while literate programming [35, 36, 44, 47, 52, 59, 67] is encouraged, I am currently not following it due to the tedious process of developing software using literate programming. Hence, a short development time for well-commented, functionally correct, and efficient source code is prioritized over code written according to the literate programming approach.

5 Exception Safety

When developing software using programming/scripting languages that enable exceptions or errors to be thrown and caught, adopt "a set of contractual guidelines" [74] to support exception/error management. This "set of contractual guidelines" is based on exception safety guarantees in *C++* [1, 2, 74] [73, Subsection §4.4 on "Writing exception safe code"].

The levels of exception/error safety listed in descending order of safety guarantees are [1, 2, 73, 74]:

1. no throw guarantee, or failure transparency: "Best level of exception safety."
2. strong exception safety, commit/rollback semantics, or no-change guarantee
3. basic exception safety
4. minimal exception safety, no-leak guarantee
5. no exception safety: "No guarantees are made. (Worst level of exception safety)"

These aforementioned levels of exception/error safety can be partially handled. Also, the use of guards is strongly recommended for making the software and library (or, circuit or system) exception safe.

Please judiciously consider what to do with the semipredicate problem [77].

6 Suggested Software Architecture

At the software system level, the software architecture can be described by the following modules/components:

1. utilities:
   (a) parser(s):
      i. For input benchmarks
   (b) output generator(s)
   (c) flag/switch -based printing information to standard output/error:

i. Print statements only when debugging mode is on.
ii. Else, squelch print/trace statements to speed up computation/performance.

2. solvers:
   (a) ODE solver(s) for ordinary differential equations (ODEs):
      i. ODE solver(s) for nonlinear ODEs.
   (b) PDE solver(s) for partial differential equations (PDEs):
      i. PDE solver(s) for nonlinear PDEs.
   (c) satisfiability modulo theories (SMT) solver(s)
   (d) boolean/proposition satisfiability (SAT) solver(s)
   (e) maximum satisfiability modulo theories (Max-SMT) solver(s)
   (f) maximum satisfiability (Max-SAT) solver(s)
   (g) pseudo-boolean optimization (PBO) solver(s)
   (h) quadratic unconstrained binary optimization (QUBO) solver(s)
   (i) weighted boolean optimization (WBO) solver(s)
   (j) framework for algorithmic portfolio optimization
3. data structures:
   (a) directed graphs:
      i. directed acyclic graphs (DAGs)
      ii. binary decision diagrams (BDDs)
      iii. AND-inverter graphs (AIGs)
   (b) undirected graphs:
      i. heaps
      ii. trees
   (c) maps, dictionaries, and hash tables
4. graphical user interface (GUI), if required.

Lastly, suggestions are not available for digital and mixed-signal integrated circuits (ICs) and VLSI systems, such as system-on-chips (SoCs). More work needs to be done in terms of looking at hardware refactoring, and hardware design patterns.

7 Additional Guidelines

Please kindly use the *Markdown* language for writing text documents. This is because Bitbucket will treat my text file as a file written in the *Markdown* syntax. That said, the raw file looks a lot better than the represented *Markdown* files. Their (`Bitbucket`) formatting for *Markdown* is messed up. *GitHub*'s formatting for *Markdown* works as expected.

In addition, tools for working with source code and LaTeX source files include:

1. `git`: [18]
2. `latexdiff`: "determine and markup differences between two latex files"
   (a) Evan Driscoll, "Latexdiff notes," from *Evan Driscoll's web page: Writings on Software: LaTeX*, the Department of Computer Sciences, University of Wisconsin-Madison College of Engineering, University of Wisconsin-Madison, Madison, WI. Available online at: http://pages.cs.wisc.edu/~driscoll/software/latex/latexdiff.html; last accessed on February 15, 2016 [16].
3. documentation generators:

   (a) `Doxygen` [70]
   (b) `Texinfo`-based generators [64–66, 78]
4. Build automation:
   (a) `SCons` [17]

Data sets and sets of benchmarks for experiments shall be publicly published using an online repository, via *figshare LLP* [21] and/or *DataHub* [15]. For each data set, or each set of benchmarks, create a unique Digital Object Identifier (DOI) [32] to identify it.

Repositories for software as well as designs of integrated circuits and cyber-physical systems shall be stored online, using online repositories such as *GitHub* [25]. Each repository shall have a unique DOI to identify it, and include all source code, documentation, and design files. There also exists cloud-based repositories for the source code of software/hardware projects that allow me to execute my software (or simulate my hardware). E.g., see [12, 58] as examples to facilitate research reproducibility, replicability, and repeatability.

Please kindly note that *GitHub* [25]:

1. Does not allow a *GitHub*-based page to be refreshed/reloaded many times in a few seconds. Else, it would report the following: x
   (a) "Whoa there!"
   (b) "You have triggered an abuse detection mechanism."
   (c) "Please wait a few minutes before you try again."

If possible, develop software in a *Python*ic style [24, 33, 54, 56, 57, 71].

References

[1] David Abrahams. Exception-safety in generic components: Lessons learned from specifying exception-safety for the C++ standard library. In Proceedings of the International Seminar on Generic Programming, volume 1766 of Lecture Notes in Computer Science, pages 69–79, Wadern, Merzig-Wadern, Saarland, Germany, April 27 – May 1 1998. Springer-Verlag Berlin Heidelberg.

[2] David Abrahams. Exception-safety in generic components: Lessons learned from specifying exception-safety for the C++ standard library. Available online from *Boost C++ Libraries: Community* at: http://www.boost.org/community/exception_safety.html; self-published; October 27, 2016 was the last accessed date, 2001.

[3] ACM Council. ACM code of ethics and professional conduct. Available online at: http://www.acm.org/about/code-of-ethics; September 27, 2014 was the last accessed date, October 16 1992.

[4] Michael Barr. Embedded C coding standard. Self-published, Gaithersburg, MD, 2013.

[5] Wojciech Basalaj and Richard Corden. High integrity C++ coding standard V4.0: An overview. Technical report, Programming Research Ltd., Hersham, Surrey, England, U.K., November 2013.

[6] Lionel Bening and Harry Foster. Principles of Verifiable RTL Design: A Functional Coding Style Supporting Verification Processes in Verilog. Kluwer Academic Publishers, New York, NY, 2000.

[7] Lionel Bening and Harry Foster. Principles of Verifiable RTL Design: A functional coding style supporting verification processes in Verilog. Kluwer Academic Publishers, New York, NY, second edition, 2001.

[8] Peter A. Blume. The LabVIEW Style Book. Prentice Hall, Upper Saddle River, NJ, 2007.

[9] Field Cady. The Data Science Handbook. John Wiley & Sons, Hoboken, NJ, 2017.

[10] Tom Cargill. C++ Programming Style. Addison-Wesley, Reading, MA, 1992.

[11] Frank M. Carrano. Data Structures and Abstractions with Java. Prentice Hall, Upper Saddle River, NJ, third edition, 2012.

[12] Code Ocean staff. Code ocean: Discover & run scientific code. Available online at: https://codeocean.com; February 2, 2018 was the last accessed date, 2017.

[13] Hewlett-Packard Company. Dave packard's 11 simple rules. Available online in *Hewlett-Packard Company: Retiree Home: History: Founders & early contributors–Dave Packard* at: http://www.hp.com/retiree/history/founders/packard/11rules.html; March 15, 2014 was the last accessed date, 2012.

[14] Jon Conway and Steve Watts. A Software Engineering Approach to LabVIEW. National Instruments Virtual Instrumentation Series. Prentice Hall, Upper Saddle River, NJ, 2003.

[15] Datopian (Atomatic Ltd), Inc. staff. DataHub: Frictionless data, data online made simple. Available online at: http://datahub.io; February 2, 2018 was the last accessed date, 2017.

[16] Evan Driscoll. Latexdiff notes. Available online from *University of Wisconsin-Madison: University of Wisconsin-Madison College of Engineering: Department of Computer Sciences: Evan Driscoll's web page: Writings on Software: LaTeX* at: http://pages.cs.wisc.edu/~driscoll/software/latex/latexdiff.html; February 15, 2016 was the last accessed date.

[17] Evan Driscoll. Scons documentation recommendatitons. Available online from *Evan Driscoll's web page: Writings on Software: SCons*, the Department of Computer Sciences, University of Wisconsin–Madison College of Engineering, University of Wisconsin–Madison at: http://pages.cs.wisc.edu/~driscoll/software/scons/index.html; February 15, 2016 was the last accessed date.

[18] Evan Driscoll. Version control. Available online from *Evan Driscoll's web page: Writings on Software*, the Department of Computer Sciences, University of Wisconsin–Madison College of Engineering, University of Wisconsin–Madison at: http://pages.cs.wisc.edu/~driscoll/software/vcs/; February 15, 2016 was the last accessed date, February 22 2012.

[19] Benny Evangelista. Facebook's hacker way – "move fast and break things". Available online in *The San Francisco Chronicle: The SF Gate: Blogs at SFGate.com: The Technology Chronicles* at: http://blog.sfgate.com/techchron/2012/02/01/facebooks-hacker-way-move-fast-and-break-things/; October 9, 2014 was the last accessed date, February 1 2012.

[20] Michael C. Feathers. Working Effectively with Legacy Code. Robert C. Martin Series. Pearson Education, Upper Saddle River, NJ, 2005.

[21] figshare LLP staff. figshare: credit for all your research. Available online at: https://figshare.com/; November 4, 2016 was the last accessed date, 2016.

[22] Michael Fingeroff. High-Level Synthesis Blue Book. Xlibris Corporation, Bloomington, IN, 2010.

[23] Rachel Fong. Move fast and break things. Available online in *MIT Admissions: Blogs: Academics & Research* at: http://mitadmissions.org/blogs/entry/move_fast_and_break_things; October 9, 2014 was the last accessed date, June 25 2011.

[24] Daniel França. C++ has become more Pythonic-Jeff Preshing. Available online from *Standard C++: Blog* at: https://isocpp.org/blog/2014/12/c-has-become-more-pythonic; self-published; February 1, 2018 was the last accessed date, December 2 2014.

[25] GitHub staff. GitHub. Available online at: https://github.com/; November 4, 2016 was the last accessed date, 2016.

[26] Todd Hoff. C++ coding standard. Available online from *Possibility Outpost: Programmer's Corner* at: http://www.possibility.com/Cpp/CppCodingStandard.html; February 16, 2016 was the last accessed date, March 1 2008.

[27] IEEE Board of Directors. IEEE code of ethics. Available online at: http://www.ieee.org/about/ieee_code_of_conduct.pdf; October 2, 2014 was the last accessed date, June 2014.

[28] IEEE Board of Directors. IEEE code of ethics. In IEEE Policies, page 7.3. Institute of Electrical and Electronics Engineers, New York, NY, 2014.

[29] IEEE Board of Directors. IEEE policies. Technical report, Institute of Electrical and Electronics Engineers, New York, NY, August 2014.

[30] IEEE Board of Directors. Nondiscrimination policy. In IEEE Policies, page 9.8. Institute of Electrical and Electronics Engineers, New York, NY, 2014.

[31] Intel Corporation staff. Intel 64 and IA-32 Architectures Optimization Reference Manual. Intel Corporation, Santa Clara, CA, September 2015.

[32] International DOI Foundation staff. Digital object identifier system. Available online at: https://www.doi.org/; January 20, 2017 was the last accessed date, January 10 2017.

[33] Jon. What does pythonic mean? [closed]. Available online from *Stack Exchange Inc.: Stack Overflow: Questions* at: https://stackoverflow.com/questions/25011078/what-does-pythonic-mean; February 1, 2018 was the last accessed date, July 29 2014.

[34] Brian W. Kernighan and P. J. Plauger. The Elements of Programming Style. McGraw-Hill, New York, NY, second edition, 1982.

[35] Donald E. Knuth. Literate programming. The Computer Journal, 27(2):97–111, 1984.

[36] Donald E. Knuth. Literate Programming. Center for the Study of Language and Information – Lecture Notes. The University of Chicago Press, Chicago, IL, 1992.

[37] Philip Koopman. Better Embedded System Software. Drumnadrochit Education, 2010.

[38] Helmut Kopka and Patrick W. Daly. Guide to LaTeX. Addison-Wesley Series on Tools and Techniques for Computer Typesetting. Addison-Wesley, Boston, MA, fourth edition, 2004.

[39] David Kushner. Facebook philosophy: Move fast and break things – hacker culture is alive and well at Facebook. Available online in *IEEE Spectrum* at: http://spectrum.ieee.org/at-work/innovation/facebook-philosophy-move-fast-and-break-things/0; October 9, 2014 was the last accessed date, June 1 2011.

[40] Jean J. Labrosse. Embedded Systems Building Blocks: Complete and Ready-to-Use Modules in C. CMP Books, Lawrence, KS, second edition, 1999.

[41] Phillip A. Laplante and Seppo J. Ovaska. Real-Time Systems Design and Analysis: Tools for the Practitioner. John Wiley & Sons, Hoboken, NJ, fourth edition, 2012.

[42] Fred Long, Dhruv Mohindra, Robert C. Seacord, Dean F. Sutherland, and David Svoboda. Java Coding Guidelines: 75 Recommendations for Reliable and Secure Programs. The SEI Series in Software Engineering. Addison-Wesley, Upper Saddle River, NJ, 1995.

[43] Ian Macdonald. The unofficial Ruby usage guide. Available online from *Caliban – Opinion and Righteous Anger: Ruby Projects: Recommended Tutorials* at: http://www.caliban.org/ruby/rubyguide.shtml; self-published; October 20, 2015 was the last accessed date.

[44] Steve McConnell. Code Complete: A Practical Handbook of Software Construction. Microsoft Press, Redmond, WA, second edition, 2004.

[45] Mike Mintz and Robert Ekendahl. Hardware Verification with *SystemVerilog*: An Object-Oriented Framework. Springer Science+Business Media, LCC, New York, NY, 2007.

[46] Trevor Misfeldt, Gregory Bumgardner, and Andrew Gray. The Elements of C++ Style. Cambridge University Press, New York, NY, 2004.

[47] Matthias Müller-Hannemann and Stefan Schirra. Algorithm Engineering: Bridging the Gap between Algorithm Theory and Practice, volume 5971 of Lecture Notes in Computer Science. Springer-Verlag Berlin Heidelberg, Heidelberg, Germany, 2010.

[48] Oracle Corporation staff. How to write Doc comments for the Javadoc tool. Available online from *Oracle Corporation: Oracle Technology Network: Recently Published Technical Articles: Articles About Java Technology* at: http://www.oracle.com/technetwork/articles/java/index-137868.html; January 30, 2017 was the last accessed date.

[49] Oracle Corporation staff. Javadoc tool. Available online from *Oracle Corporation: Oracle Technology Network: Recently Published Technical Articles: Articles About Java Technology* at: http://www.oracle.com/technetwork/articles/java/index-jsp-135444.html; January 30, 2017 was the last accessed date.

[50] Oracle Corporation staff. Javadoc tool. Available online from *Oracle Corporation: Oracle Technology Network: Oracle Technology Network for Java Developers: Java SE at a Glance: Java SE Documentation* at: http://www.oracle.com/technetwork/java/javase/documentation/index-jsp-135444.html; January 30, 2017 was the last accessed date.

[51] Oracle Corporation staff. javadoc. Available online from *Oracle Corporation: Oracle Help Center: Java Platform, Standard Edition (Java SE) 8: Java Platform Standard Edition 8 Documentation: JDK Tools and Utilities: Java Platform, Standard Edition Tools Reference: §5 Create and Build Applications* at: https://docs.oracle.com/javase/8/docs/technotes/tools/windows/javadoc.html; January 30, 2017 was the last accessed date, January 2016.

[52] Andy Oram and Greg Wilson. Beautiful Code: Leading Programmers Explain How They Think. O'Reilly Media, Sebastopol, CA, 2007.

[53] Oren Patashnik. BibTeXing. Available online at: http://mirrors.ctan.org/biblio/bibtex/base/btxdoc.pdf; September 24, 2014 was the last accessed date, February 8 1988.

[54] Jeff Preshing. C++ has become more Pythonic. Available online from *Preshing on Programming* at: http://preshing.com/20141202/cpp-has-become-more-pythonic/; February 1, 2018 was the last accessed date, December 2 2014.

[55] Programming Research Ltd. staff. High integrity C++: Coding standard version 4.0. Technical report, Programming Research Ltd., Hersham, Surrey, England, U.K., October 3 2013.

[56] Kenneth Reitz. Code style. Available online from *The Hitchhiker's Guide to Python* at: http://docs.python-guide.org/en/latest/writing/style/; self-published; February 1, 2018 was the last accessed date, 2016.

[57] Kenneth Reitz and Tanya Schlusser. The Hitchhiker's Guide to Python: Best Practices for Development. O'Reilly Media, Sebastopol, CA, 2016.

[58] RunMyCode Association members. RunMyCode. Available online at: http://www.runmycode.org/; November 13, 2016 was the last accessed date, 2013.

[59] Stephen R. Schach. Object-Oriented and Classical Software Engineering. McGraw-Hill, New York, NY, seventh edition, 2007.

[60] Anna Schneider. How to become a data scientist before you graduate. Berkeley Science Review, July 30 2013.

[61] Science Infusion Software Engineering Process Group staff. Science infusion software engineering process group (SISEPG) C++ programming standards and guidelines. Available online as Version 1.11 from *United States Department of Commerce: National Oceanic and Atmospheric Administration: National Weather Service: Office of Hydrologic Development, now known as the National Water Center: Hydrology Laboratory* at: http://www.nws.noaa.gov/oh/hrl/developers_docs/C++_Software_Standards.pdf; October 27, 2015 was the last accessed date, November 17 2006.

[62] James Shore and Shane Warden. The Art of Agile Development. Theory in Practice. O'Reilly Media, Sebastopol, CA, 2008.

[63] Kevin A. Smith and Doug Kramer. Requirements for writing Java API specifications. Available online from *Oracle Corporation: Oracle Technology Network: Oracle Technology Network for Java Developers: Java SE at a Glance: Java SE Documentation* at: http://www.oracle.com/technetwork/java/javase/documentation/index-142372.html; January 30, 2017 was the last accessed date, January 2003.

[64] Richard Stallman and Bob Chassell. GNU Texinfo 6.3. Available online from *The GNU Operating System and the Free Software Movement: GNU Software: Texinfo – The GNU Documentation System: Texinfo manuals: GNU Texinfo manual* at: https://www.gnu.org/software/texinfo/manual/texinfo/texinfo.html; November 4, 2016 was the last accessed date, September 11 2016.

[65] Richard Stallman and Bob Chassell. GNU Texinfo manual. Available online from *The GNU Operating System and the Free Software Movement: GNU Software: Texinfo – The GNU Documentation System: Texinfo manuals* at: https://www.gnu.org/software/texinfo/manual/texinfo/; November 4, 2016 was the last accessed date, September 11 2016.

[66] Richard Stallman and Bob Chassell. Texinfo: The GNU documentation system. Available online from *The GNU Operating System and the Free Software Movement: GNU Software* at: https://www.gnu.org/software/texinfo/; November 4, 2016 was the last accessed date, September 11 2016.

[67] Venkat Subramaniam and Andy Hunt. Practices of an Agile Developer: Working in the Real World. The Pragmatic Programmers, Raleigh, NC, 2006.

[68] Eno Thereska. C++ coding standard. Available online from *Eno Thereska's home page: Department of Electrical and Computer Engineering, College of Engineering, Carnegie Mellon University* at: https://users.ece.cmu.edu/~eno/coding/CppCodingStandard.html; February 16, 2016 was the last accessed date.

[69] Jonathan W. Valvano. Embedded Microcomputer Systems: Real Time Interfacing. Cengage Learning, Stamford, CT, third edition, 2007.

[70] Dimitri van Heesch. Doxygen. Available online at M.C.G.V. Stack: http://www.stack.nl/~dimitri/doxygen/ and http://doxygen.org/; November 4, 2016 was the last accessed date, September 11 2016.

[71] Guido van Rossum, Barry Warsaw, and Nick Coghlan. Style guide for Python code. Available online as *Python: Python Developer's Guide: Python Enhancement Proposals: Python Enhancement Proposal 0* at: https://www.python.org/dev/peps/pep-0008/; October 20, 2015 was the last accessed date, August 1 2013.

[72] John L. Weatherwax. Solution manuals and various book notes. Available online at: http://waxworksmath.com/ce_solutionmanuals.asp; self-published; May 31, 2015 was the last accessed date, 2008.

[73] Wikibooks contributors. Exception handling. In C++ Programming, chapter 4. Wikimedia Foundation, San Francisco, CA, February 11 2016.

[74] Wikipedia contributors. Exception safety. Available online from *Wikipedia, The Free Encyclopedia: CATEGORY* at: https://en.wikipedia.org/wiki/Exception_safety; October 29, 2016 was the last accessed date, September 26 2016.

[75] Wikipedia contributors. Indent style. Available online in *Wikipedia, The Free Encyclopedia: Source code* at: https://en.wikipedia.org/wiki/Indent_style; November 4, 2016 was the last accessed date, October 30 2016.

[76] Wikipedia contributors. Natural language programming. Available online in *Wikipedia, The Free Encyclopedia: Computer programming* at: https://en.wikipedia.org/wiki/Natural_language_programming; November 4, 2016 was the last accessed date, November 2 2016.

[77] Wikipedia contributors. Semipredicate problem. Available online from *Wikipedia, The Free Encyclopedia: Programming language topics* at: https://en.wikipedia.org/wiki/Semipredicate_problem; October 29, 2016 was the last accessed date, October 23 2016.

[78] Wikipedia contributors. Texinfo. Available online in *Wikipedia, The Free Encyclopedia: Markup languages* at: https://en.wikipedia.org/wiki/Texinfo; November 4, 2016 was the last accessed date, April 29 2016.

[79] Wikipedia contributors. Naming convention (programming). Available online from *Wikipedia, The Free Encyclopedia: Source code* at: https://en.wikipedia.org/wiki/Naming_convention_(programming); March 10, 2017 was the last accessed date, February 26 2017.

[80] Clifford Wolf. Clifford on programming style. Available online from *Clifford Wolf's Personal Homepage* at: http://www.clifford.at/style.html; self-published; March 14, 2016 was the last accessed date.

[81] Mark Zuckerberg. Letter from Mark Zuckerberg. Available online as *Form S-1 REGISTRATION STATEMENT Under The Securities Act of 1933, Facebook, Inc.* at: http://www.sec.gov/Archives/edgar/data/1326801/000119312512034517/d287954ds1.htm#toc287954_10; October 9, 2014 was the last accessed date, February 1 2012.