Guidelines for Collaboration

Zhiyang Ong *

January 25, 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* [**?**, **?**, **?**, **?**]; sharing of source code, design files, sets of benchmarks, data sets, and documentation on online repositories [**?**, **?**]; 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.

*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)` [**?**, **?**, **?**] and the `Association for Computing Machinery (ACM)` [**?**]. Also, actions of discrimination are not acceptable [**?**]; we should intentionally commit to inclusive diversity. An additional guideline is "Dave Packard's 11 simple rules" [**?**].

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 [**?**, **?**]. Also, when considerable effort has been invested in an automated regression testing/verification infrastructure, do not be afraid to "move fast and break things" [**?**, **?**].

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 [**?**] for a list of BibTeX entry types; alternatively, refer to the *Wikipedia* entry for , or [**?**, §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), 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 [**?**], *Javadoc*-based 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*) [**?**].

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) [**?**].

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.

For a suggested coding style for `Python` and *Ruby* scripts, see [**?**] and [**?**], respectively.

While well-documented source code is desired, natural language programming [**?**] is usually infeasible due to the choices of programming/computer languages used. Also, while literate programming [**?**, **?**, **?**, **?**, **?**, **?**, **?**] 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" [**?**] to support exception/error management. This "set of contractual guidelines" is based on exception safety guarantees in *C++* [**?**, **?**, **?**] [**?**, Subsection §4.4 on "Writing exception safe code"].

The levels of exception/error safety listed in descending order of safety guarantees are [**?**, **?**, **?**, **?**]:

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 [**?**].

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`: [**?**]
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 [**?**].
3. documentation generators:
    (a) *Doxygen* [**?**]
    (b) `Texinfo`-based generators [**?**, **?**, **?**, **?**]:
        i.
4. Build automation:
    (a) `SCons` [**?**]

Data sets and sets of benchmarks for experiments shall be publicly published using an online repository, via *figshare LLP* [**?**]. For each data set, or each set of benchmarks, create a unique Digital Object Identifier (DOI) 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* [**?**]. Each repository shall have a unique DOI to identify it, and include all source code, documentation, and design files.

Please kindly note that *GitHub* [**?**]:

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