J Mol Model (2003) 9:271–272
DOI 10.1007/s00894-003-0141-1

ORIGINAL PAPER

François-Yves Dupradeau · Jacques Rochette

Bugs in computational chemistry software and their consequences: the importance of the source code

Received: 25 February 2003 / Accepted: 16 May 2003 / Published online: 11 July 2003

Abstract The increase in computer power and the development of new mathematical concepts implemented in software have allowed computational chemistry to emerge as a new research field. Although programs were freely distributed during the "golden age" of this discipline, today they are usually copyrighted and have become easier and easier to use through sophisticated graphical interfaces. This "democratization" is a vector of success for this discipline. Nowadays, non-theoreticians can use such programs more easily and solve chemistry-related problems with the computer. The number of program offerings has rapidly grown and private companies specialized in molecular modeling have appeared and compete to sell their products. Thus, numerous software packages, often presenting similar capabilities, are now available on the market. Within this context, the availability of the program source code remains, in our opinion, an important criterion for program selection.

Keywords Academic, Free, Open source and proprietary software · Open versus closed source code · Software bug

Bugs and software: general considerations

A bug is defined as an error or a defect that causes a program to malfunction. All programs have bugs and they represent a nightmare for programmers and users. This was exemplified in the past by the bug of the millennium, the Y2K bug. It was responsible for a global fear and was even named TEOTWAWKI, i.e., the End of the World as We Know It. [1] It is unrealistic to attempt to write bug-free software and to find all the program errors. Indeed, for big and multiplatform programs, so many combinations of parameters can be found that it is impossible to test all of them before the program is released. For this reason, bugs have to be considered as a part of the program. Users and developers have learned how to work with programs and their potential errors, which can show up at any time. Furthermore, a program is generally protected by a disclaimer implying that the results obtained with the program cannot be guaranteed.

Concerning computational chemistry and more generally scientific software, specific problems inherent in the software type must be underlined. Computational chemistry programs are usually written by physicists, chemists and programmers while some users might be non-specialists running them as "black boxes", meaning without fully understanding the algorithms behind them. Numerous bugs make a program crash and prevent the user from realizing a task. Although this is a real nuisance, bugs generating improper results are the worst since they are difficult to detect. Therefore, the consequences of bugs in scientific software are far more damaging than those in word-processing type programs, as erroneous results may be produced, trusted and even published.

"Free", "open source" and "academic" software

Although "free" and "open source" software belongs to two different movements for semantic reasons, both communities defend a similar philosophy. [2, 3] The basic idea behind this is that each user has the freedom to read, execute, modify and redistribute the source code. Therefore, a prerequisite of "free" and "open source" software is that the sources be available to everyone. "Academic" computational chemistry software such as TINKER, [4] AMBER, [5] CHARMM, [6] WHATIF [7] or GAMESS [8] (among many others) do not follow exactly the same rules. Indeed, such programs are usually copyrighted and

F.-Y. Dupradeau (✉)
GRBPD, UPRES EA 2629, Faculté de Pharmacie,
Université de Picardie Jules Verne,
1–3 rue des Louvels, 80037 Amiens, Cedex 1, France
e-mail: fyd@u-picardie.fr
Tel.: +33-(0)3-2282-7494

J. Rochette
GRBPD, UPRES EA 2629, Faculté de Médecine,
Université de Picardie Jules Verne,
1–3 rue des Louvels, 80037 Amiens, Cedex 1, France

their license contains restrictions intended to preserve notices of authorship and development control. However, “free”, “open source” and “academic” software share the same crucial characteristic: the source code is provided.

The reliability of “free” and commercial UNIX software has been studied and compared. Miller et al. tested a large collection of UNIX programs. [9] They showed that GNU programs are more reliable and noticeably better than commercially produced software. GNU developers explain that it is not an accident that their programs are of such high quality. [10] An author making the source code available for everyone had better write a clear and clean source code, knowing his/her reputation is at stake. Moreover, as the sources are available, the whole community becomes involved in correcting problems and sharing the improvements in total transparency. It is clear that the work of testing and debugging performed by a whole community from around the world will be more efficient than that done by only a few programmers, keeping the source code hidden. This universal collaboration raises the quality level of the program and should be the ultimate goal in scientific software. Thus, the source code is a pledge of reliability and development. Otherwise, the program could vegetate and end up trapped as a fossil. This is well illustrated by the logo of the AMBER software where two citations have been successively used: “All the bugs are not out of AMBER” and “Don’t stay trapped in fossil AMBER”. [11, 12]

Commercial and proprietary software

Nowadays, a great number of private companies, which do not share the same goals, sell commercial and proprietary computational chemistry software. The source code of their software is generally not provided and the program is only distributed in the executable format, often at a considerable cost. In most cases, this strategy is applied in order to protect the program from unauthorized use. It is well known that this type of software has one advantage over “academic” software in that it is more functional because of elaborate molecular graphics features and because it is better documented and supported. However, since the sources of such software are not available, modelers have no other choice than to use them as “black boxes”. This can be a significant limitation especially in “academic” research where the necessity of studying the implementation of algorithms and/or of modifying the source code of the program might be important. Moreover, bugs remain a mystery to the community and consequently their corrections are fully controlled by the provider. Once a bug is uncovered, the users are then faced with having to wait for the company to patch up the problem. In the end, the bug may be corrected at a certain cost or worse, never be corrected at all. This means that a new program has to be acquired or developed which represents a considerable expense.

This viewpoint clearly reveals how important it is to have access to the program source code in the computational chemistry field and more generally in scientific software. This has been confirmed by Gaussian Inc., [13] which not only sells the binary of their Gaussian 03 program, [14] but also gives their clients the opportunity to buy the source code. Thus, in our opinion, the availability of the source code, a guarantee of reliability and evolution for a program, must be considered as a significant criterion for selecting software. More particularly, modelers who begin in computational chemistry are not always aware of this open–closed source duality and should not consider only functionality when making their choice. We hope this will encourage researchers from private and “academic” circles not only to use “open” source software but to distribute them as well.

References

1. Grossman WM (1998) Sci Am October:48
2. http://www.gnu.org/
3. http://www.opensource.org/
4. http://dasher.wustl.edu/tinker/
5. Pearlman DA, Case DA, Caldwell JW, Ross WS, Cheatham III TE, DeBolt S, Ferguson D, Seibel G, Kollman PA (1995) Comput Phys Commun 91:1–41
6. Brooks BR, Bruccoleri RE, Olafson BD, States DJ, Swaminathan S, Karplus M (1983) J Comput Chem 4:187–217
7. Vriend G (1990) J Mol Graph 8:52–56
8. Schmidt MW, Baldridge KK, Boatz JA, Elbert ST, Gordon MS, Jensen JH, Koseki S, Matsunaga N, Nguyen KA, Su S, Windus TL, Dupuis M, Montgomery J (1993) J Comput Chem 14:1347–1363
9. Miller BP, Koski D, Lee CP, Maganty V, Murthy R, Natarajan A, Steidl J (1995) Fuzz revisited: a re-examination of the reliability of UNIX utilities and services. In: Computer sciences technical report #1268. University of Wisconsin-Madison, April
10. http://www.gnu.org/software/reliability.html
11. http://amber.scripps.edu
12. http://amber.scripps.edu/bugfixes.html
13. Gaussian, Carnegie Office Park, Building 6, Suite 230, Carnegie, Pa. 15106, USA, http://www.gaussian.com
14. Frisch MJ, Trucks GW, Schlegel HB, Scuseria GE, Robb MA, Cheeseman JR, Montgomery Jr JA, Vreven T, Kudin KN, Burant JC, Millam JM, Iyengar SS, Tomasi J, Barone V, Mennucci B, Cossi M, Scalmani G, Rega N, Petersson GA, Nakatsuji H, Hada M, Ehara M, Toyota K, Fukuda R, Hasegawa J, Ishida M, Nakajima T, Honda Y, Kitao O, Nakai H, Klene M, Li X, Knox JE, Hratchian HP, Cross JB, Adamo C, Jaramillo J, Gomperts R, Stratmann RE, Yazyev O, Austin AJ, Cammi R, Pomelli C, Ochterski JW, Ayala PY, Morokuma K, Voth GA, Salvador P, Dannenberg JJ, Zakrzewski VG, Dapprich S, Daniels AD, Strain MC, Farkas O, Malick DK, Rabuck AD, Raghavachari K, Foresman JB, Ortiz JV, Cui Q, Baboul AG, Clifford S, Cioslowski J, Stefanov BB, Liu G, Liashenko A, Piskorz P, Komaromi I, Martin RL, Fox DJ, Keith T, Al-Laham MA, Peng CY, Nanayakkara A, Challacombe M, Gill PMW, Johnson B, Chen W, Wong MW, Gonzalez C, Pople JA (2003) Gaussian 03, Revision A.1. Gaussian, Pittsburgh, Pa.