

# Multibody Software Accessibility

## An Overview and Some Considerations

Pierangelo Masarati

Novembre 19, 2007

### Abstract

This work describes what are the generic implications of software access conditions and policies and, significantly, what their impact can be on general-purpose multibody software, highlighting the advantages and the drawbacks they have with respect to researchers both as end users and software developers.

**Keywords:** Software Access, Software Licensing, Free Software, Open Source Software, Multibody Software.

## 1 Foreword

This work might not be considered truly “scientific”, essentially because it addresses issues related to software access policies and software development philosophies, which are more contiguous to social sciences than to engineering or computer science. Furthermore, the author received little education, if any, in those areas, including computer science itself.

However, the considerations presented in this paper rely on a more than 10-year-long experience in free software development, both as a researcher (in aerospace engineering and multibody dynamics) and as a professional software developer (in different fields, including, but not limited to, multibody dynamics and engineering software in general).

The aim of this work is to convey the importance of accessibility to software as end-users and potentially as developers, and to propose a categorization of software, based on access policy, that goes beyond the traditional, possibly pointless subdivision between commercial and academic software.

The work is not based on a really scientific approach, but rather on extensive searches in the Internet (“googling”) and partly on personal experience and direct knowledge of some of the software discussed in the following, and of their developers.

In fact, it is the author’s belief that scientific software use, and scientific education supported by software use, is tightly related to the level of access to the software that is used; of course, the more the access, the better the use

and the education. As a consequence, software categorization based on access rights is an important parameter (the most important, in the author's opinion) in driving the selection of scientific software.

## 2 Introduction

Classifying things (objects, ideas) into categories has always been part of our approach to comprehension, beginning from the ancient Greek philosophers.

This work is of course much less ambitious, since it only aims at proposing modifications to commonly accepted software categorizations, especially outside the world of Information Technology (IT), and moreover restricting the field to multibody dynamics simulation software.

Usually categorization cannot be considered separately from the use one intends to make of it. So the intended use of the presented categorization is stated from the beginning, to avoid any confusion for the reader (and to allow readers to stop reading, in case this relatively uncommon topic does not match their expectations).

It is the author's intention to help the reader determine what criteria should be considered when selecting software for scientific use, and how to discriminate between the existing literature, categorizations and best practice, in order to clarify what is important and why.

Of course this work reflects the ideas of the author on software development philosophy and on software access policy, so it might be sometimes biased toward one philosophy and one policy (to make it clear: those of free software). References to other types of software development philosophies and software access policies, however, are given trying to be as objective as possible, and based on information coming from the developers and/or the distributors of those software (those who replied to the author's requests, at least).

Finally, it is not the author's intent to judge what policy is best. The reader will hopefully be able to judge. It is worth stressing that this paper is not aimed at putting all the blame on commercial software developers, as they probably do their job the best they can, and they usually do a great job, indeed.

## 3 Software Categorization

### 3.1 Beyond "Commercial" vs. "Academic" Software

The software classification that is often informally used, especially when scientific software is concerned, is between "commercial" and "academic" software. Let us have a closer look at the meaning we usually attribute to these two terms.

Commercial software is usually perceived as that type of software that comes only in non-human readable form, with some commercial support: a salesperson, a phone number to call in case of trouble (or just a mail address or a website, if you are not their best customer), captivating theory and user manuals. Moreover, nowadays commercial software always have a series of exciting things like

users' conferences, usually held in charming locations, and lots of other fringes, including significant discounts for universities.

On the other side, we often see the academic software. Academic software is what we academicians usually work at. At the bottom line, it is a Proof-Of-Concept (POC) code, in many cases implemented, under the supervision of a Professor, by one or more students whose main goal consists in graduating as soon as possible. So either the Professor works most of the time side by side with the student, or that piece of academic software (or, academic piece of software, as you prefer) is lost forever. To cut a long story short, academic software is often the poor version of so-called commercial software: usually it can do few brilliant things, provided the problem is simple enough, but it is mainly aimed at allowing the developers to publish few (hopefully) brilliant papers (or perish).

The point here is that there is no intellectual difference between so-called commercial and academic software: commercial software has a theory manual, while academic software is based on brilliant scientific papers, usually accessible only by purchasing scientific papers. So the theory they are based on is accessible to a similar level. Apparently, they mainly differ in that commercial software is usually much more useful than its academic counterpart, because its main purpose *is to be useful*, and thus provides commercial grade user interfaces and user support, which the latter usually does not, but in theory both provide an end user the same level of access (and usually the so-called commercial software makes access much easier).

## 3.2 Free Software

What really makes the difference, here, is the concept of freedom. If software were free, in the sense that anyone could really access it in the manner that best suits their needs, all players would earn something, and thus all players would win. If the difference between so-called commercial and academic software comes down to mere usefulness, let us make academic software as useful, efficient and easy to use as commercial software, or even more, and then let us make it free, by making it freely accessible by others. Then others can benefit from our work and, in turn, improve it, by making it fit their needs as well.

Note that a commonly accepted definition of free software, as illustrated later, does not prevent it from being sold for a fee. So, selling free software, which is allowed by the definition of freedom, is a commercial activity. As a consequence, free software can be considered commercial software as well, without any contradiction [1].

What needs to be considered is: what often pushes academic software development? In some cases, it is the need to fulfill some requirement within research grants and contracts. In other cases, it is just the need to test new ideas and concepts, eventually meant for publication.

In the first case, if the software is part of a contract, we are basically acting like commercial software developers, with the disadvantage that usually we do not just sell our work, we also sell our rights on our work, since we commit to not selling it to our customers' competitors.

On the contrary, commercial software developers also sell their software to their customers' competitors, and the software they sell to competitors benefits from bugs fixed and enhancements added thanks to their customers. So, for a company, using commercial software that is not free means paying for a service that indirectly results in helping their competitors.

In the end the academia, by developing non-free software, is already handicapped when confronted to commercial software developers, even from a theoretical point of view. A complete change of viewpoint consists in selling expertise rather than software. Note that the academia almost always sells expertise, rather than just software; only, we tend to confuse software with expertise. To operate this change of viewpoint, the academia needs to develop the software *outside of* and *independently from* any specific grant or research contract, and then eventually fulfill the requirements of many grants and contracts using that software.

This way, the software is no longer part of the contract, and can be distributed within the contract according to its license. Of course, this approach requires the academia to invest resources (not just money, but mostly manpower) in developing software. Or, if it is already available, to exploit some free software.

But why, free? Because if it is not free it cannot be used, it cannot be adapted to the specific needs of our research, and it cannot be delivered to the customer.

Then, let us go back to the second case, where software is developed just for pure research, e.g. to prove that an algorithm works as expected. In this case, unless we are just happy to prove our brand new anything-worth-preserving, anything-not-worth-dissipating algorithm with yet-another-planar-rigid-bodies-with-no-friction-no-closed-loops problem, we would need to reinvent the wheel. Or, if it is already available, we could exploit some free software, for the very same reasons that were mentioned earlier; if our algorithm works, it could then be incorporated in that software, and would be available to others, with mutual benefit.

### 3.3 Open Source vs. Free Software

The term "Free/Libre/Open-Source Software" (FLOSS) is often associated to software whose source code is generically accessible for no fee. But the question is: can open source software<sup>1</sup> be confused with free software? This is a good question. What is the difference between open source and free software? Basically, open source is free without freedom; or, free software is open source with freedom. Why is freedom so important here? Freedom in software access is the cornerstone that guarantees the possibility to continue relying not only on our work, but also on others'.

Would anyone be happy of seeing his own work, just because it was released open source instead of free software, be taken by someone else, stuffed into some

---

<sup>1</sup>For an extensive discussion, see <http://www.opensource.org/> and [1], Chapter 6.

closed source software and sold to the rest of the world<sup>2</sup>?

Or, would anyone be happy of developing some tremendous software that relies on some broadly diffused external facility, which is open source instead of free software, and see that facility suddenly change license to some closed source form, thus preventing interoperation with future versions of that software<sup>3</sup>?

For these reasons, having access to just the source code, without real freedom, is indeed a legitimate option, possibly fine and perfectly acceptable in many cases, but it is not the preferred answer to the needs of a conscious user.

## 4 Software Accessibility

Let us focus on the end-user. Multibody software, like most software designed to assist engineering, may have a broad class of users, ranging from industry designers to university undergraduate students. Determining what access is most appropriate for each class of users may not be simple, because the members of each class can have rather different needs.

For example, in some types of industries, or for some types of products, standard software, with standard procedures, may be enough to fulfill most of the requirements. On the contrary, in other types of industries or for other types of products, software customization may represent a strong requirement.

At the same time, in the university, junior class students may just need to familiarize with software tools by modeling simple problems, while undergraduate and graduate students, working on very specific subjects for their dissertation, may need to test new elements, formulations, algorithms, and thus have very specific needs involving the use of non-standard features, which may require heavy customization of the code.

### 4.1 Access Levels

The previous discussion naturally leads to a classification of what types of access a user may need and may benefit from.

- The first fundamental access level (we will call it “level 0”) consists in the possibility of using the software to fulfill the user’s needs. The right to use a software may be conditioned by the purpose (educational, non-profit, for profit), the use of the results (some software vendors explicitly prohibit to publish results of comparative benchmarks with competing software), and the way it is used. In fact, most commercial licenses either require a user to purchase multiple licenses for multiple installations, or pose a limit on the number of concurrent executions of the software.
- The second fundamental access level (call it “level 1”) consists in the possibility to understand what the code exactly does; this, in turn, for

---

<sup>2</sup>This has been reported for the BSD code implementing TCP/IP, which was allegedly (and legally, as it is open source) used into early versions of Microsoft’s Windows operating system.

<sup>3</sup>This happened once with the X11 window system [1].

complete exploitation, requires access to the source code. There are different possibilities, for a user, to access the source code, typically under specific license restrictions that try to define the scope of the access, the purpose, and the level of confidentiality that is required to the user. An important point in accessing the code addresses the possibility to *modify* the source code, either to fix software bugs or to enhance the existing code by implementing new algorithms or improving the existing ones.

## 4.2 Access Rights

This paragraph talks about rights. Here we consider, of course, the rights of a conscious user of multibody software, which is supposed to be a well-educated person and, potentially, a developer himself. The rights of the user on a piece of software determine what level of freedom a user has with respect to that software. A description of what rights determine the freedom of a piece of software is given below, adapted from Richard Stallman's essays ([1], Chapter 3).

- The first right (we will call it “freedom 0”) is the right of using the software for any purpose one considers appropriate. Access to the software at least in binary form (“level 0”), and to its theory and use documentation is a prerequisite.
- The second right (call it “freedom 1”) is the right of modifying the software: to fix a bug, to improve it, or to make it solve a user's specific problem, which might differ from the problem the software was originally designed to solve. Access to the source code (“level 1”) is a prerequisite; access to the documentation is not, as the developer is supposed to be able to read the source code, but good documentation is usually welcome<sup>4</sup>...
- The third right (call it “freedom 2”) is the right to distribute copies of the software, either for free or for a fee, provided no rights are restricted. Again, access to the software (“level 0”) at least in binary form, and to its documentation is a prerequisite.
- The fourth right (call it “freedom 3”) is the right to distribute the software after it has been modified, provided no rights are restricted. Again, access to the source code (“level 1”) is a prerequisite.

Note that those freedoms are not negotiable: allowing the user to run a software only for non-profit use means depriving the user of freedom 0 entirely.

We can also note that access to the source code (“level 1”) is an important part of the freedoms, but it does not imply any freedom in itself, so we can state that the rest of the freedoms is at least as important, or even more (this is the author's opinion).

---

<sup>4</sup>One aspect of free software users often complain about is the lack of complete, accurate and updated documentation that affects many otherwise excellent products. This is a sad fact, unfortunately.

A quick look at Table 1 shows that usually freedom comes all or none, with few exceptions. To be fair, one should note that the table considers freedoms granted only when they are complete. In some cases, use of non-free software is allowed, and, in the case of Chrono::Engine, even access to the source code is granted, but with restrictions on the use. What makes non-free software use not appealing is the fact that efforts in using it, in developing models, and in developing the software itself in case of source code access, would be wasted in case one needs to use it for any of the purposes use is denied. For this reason, partial freedom can be even worse than no freedom, since it could trick users into spending efforts that, to pay back, need to give up all freedom at some point, to comply with non-free licenses.

Freedom 0 implies that while the software developer retains the copyright on his product, he is not entitled to restrict the way the product is used. Freedoms 2 and 3 require to distribute the software in source form as well, including modifications in case of freedom 3. This does not mean that a developer must always distribute his own modifications and enhancements; it simply means that as soon as one distributes the software in any form, the source code has to be distributed as well, and with the original rights preserved.

To clarify: if one modifies free software and does not want to distribute his own modifications in source form, the software cannot be distributed nor sold at all, in any form, but the developer, thanks to freedoms 0 and 1, can legitimately use it, even for a profit.

## 5 Ease of Access

Rights on software determine what one can do with it, which is closely related to its usefulness; however, rights alone may not be enough for a piece of software to be successful. Note, by the way, that rights have little to do with success; they are much more important. However, if a software has no success, which means it does not have enough users to justify its maintenance and development, it is inevitably destined to oblivion.

What can help in making software useful to potential users, and thus potentially successful, is the availability of some facilities. Just to mention the most important ones:

- an official web site, to provide a single point of information, contact, download and so (this usually implies a web site administrator);
- one or more mailing lists, for announcements, usage and development discussion (this usually implies a moderator, as unmoderated mailing lists often become useless because polluted by off-topic or too incompetent and inconclusive messages);
- a bug tracking system, to allow users to submit bug reports, patches, and keep track of their handling by the official developers;
- a versioning system, to allow version tracking;

- a concurrent development system, to allow multiple users to concurrently manage source code for development and bug fixing.

In some cases, some of those facilities may not be strictly required; for example, when official development is performed by just a few individuals, a concurrent development system may not be strictly required. However, a versioning system is always advisable, to ease regression tracking.

In many cases, most of those facilities do not need to be exposed to external or anonymous users. For example, read access to the source repository could be restricted, provided frequent releases occur.

In conclusion, those facilities can be key to the success of a piece of software, but they have no direct impact on access rights per se.

## 6 The IFToMM Web Site

The International Federation for the Promotion of Mechanisms and Machine Science (IFToMM) Technical Committee for Multibody Dynamics publishes a web site<sup>5</sup> (<http://193.144.52.16/>) that represents an interesting attempt to create a database of people, projects, publications, conferences and software related to multibody dynamics.

As commonly accepted most of the times, it initially classified software as “commercial” vs. “academic”; only after a discussion with the author of this work, was it agreed that a better classification would have been based on access levels.

Now the classification is based on:

- commercial software
- no-cost software
- free software

Note that this classification does not cover all options yet; in fact, it does not consider inaccessible software, basically those codes that for any reason are only accessible to their developers or sponsors (e.g. software internally developed for industrial, academic or military purposes). But probably those cases do not deserve publicity. Moreover, it does not take into account the fact that free software can be commercialized as well, much like so-called commercial software, as already discussed, but this would probably complicate things too much.

The following sections address the accessibility of the software listed at IFToMM, and the related freedoms. Results are summarized in Table 1. The reader is referred to the web site for details on each software.

---

<sup>5</sup>No friendly name, apparently; the IP resolves to `vieira.eps.cdf.udc.es`

## 6.1 So-Called “Commercial” Software

The site lists five so-called commercial software, presumably in subscription order, with a comment presumably entered by the software developers. None of them offers any freedom. They are listed in the following for informational purposes, to highlight any specific policy with respect to users’ access and freedom.

### 6.1.1 SIMPACK

The SIMPACK software is developed by INTEC GmbH. According to the web site, it allows a six weeks free test license, with restrictions.

### 6.1.2 ADAMS

The ADAMS software is developed by MSC/SOFTWARE. The company used to offer a limited version of the software for no fee, restricted to non-commercial use, but this is no longer available because of its limited usefulness, due to the fact that the software is modular and thus can be downgraded in a straightforward manner by simply omitting specific modules.

However, the company has a strong University oriented policy, as illustrated at the website <http://www.mscsoftware.com/university/>, whose mission consists in increasing the number of engineers coming out of universities who are proficient with MSC products. This can result in accessing bundles of integrated CAE software with price reductions down to 10% of the full cost, with some limitations: basically, use for commercial purposes is not allowed, and very specialistic modules might not be licensed.

### 6.1.3 DynaFlexPro

The DynaFlexPro software is an external Maple package, developed by Motion-Pro Inc. The website does not mention any special access possibilities, nor any special facilitation for non-commercial or academic use.

### 6.1.4 NEWEUL

The NEWEUL software is developed by the Universität Stuttgart. According to the web site, it offers a free demo version limited to 4 degrees of freedom.

### 6.1.5 SAMCEF Mecano

The SAMCEF Mecano software is developed by Samtech. Apparently, the website does not mention any special access possibilities, nor any special facilitation for non-commercial or academic use.

However, special access conditions are possible:

- a limited version of the software, with restrictions in the number of degrees of freedom (5000 for the FEM linear analysis, 100 for the Mecano nonlinear structural mechanics), is sold for the raw price of the CD-ROM;

- an academic version, limited in the number of working posts, is available at a very convenient price;
- research institutions can purchase the software with a special discount, provided it is not used for profit;
- a demo version, with limited duration license, can be accessed during purchase negotiations.

## 6.2 So-Called “No-Cost” Software

The site lists two no-cost software.

### 6.2.1 ROBOTRAN

The ROBOTRAN software is developed by the Université Catholique de Louvain. It is not available for download, but it can be used for free online. As a consequence, the user only has portions of freedom 0, depending on the quality of the connectivity to Louvain, and to their willingness to continue the service. In the end, the user is not granted any freedom.

### 6.2.2 SPACAR

The SPACAR software is developed by the University of Twente. The binaries can be freely downloaded and used for commercial and non-commercial purposes. However, it cannot be modified<sup>6</sup> (!) and it cannot be distributed, so it deprives the user of all freedoms except freedom 0.

## 6.3 Free Software

The site lists only one free software.

### 6.3.1 MBDyn

The MBDyn software started as an internal project at the University “Politecnico di Milano”, and since the mid 1990’s it has been mainly developed and coordinated by the author of this paper. It is now developed by the community of its users, as since 2001 it has been distributed under the GNU General Public License, and thus gives users all freedoms.

In terms of ease of access: those who are interested can gather information about it through a web site; “announce”, “users” and “devel” mailing lists are available respectively to be informed on releases, to discuss software usage and bugs, and to allow persons not directly involved in the development to discuss development strategies and details. There is no formal bug tracking system,

---

<sup>6</sup>Verbatim, the license states that: “Use of the unmodified binary (sic) distribution of the software as available from this web server is free for commercial and non-commercial use.” A restrictive interpretation implies that use of a modified binary version is not free, and thus not allowed.

as only sporadic notifications occur, and since the mailing lists are archived, they serve the purpose well enough. Versions are tracked, and concurrent development is in place, although currently restricted to authorized users only. Brave users who want to play with the latest developments can get source code snapshots by putting a query on the `mbdyn-devel@mbdyn.org` mailing list.

## 7 Other Examples

A quick search on the Internet points out plenty of other software related to multibody dynamics. Among the many links, those by Professors McPhee

`http://real.uwaterloo.ca/~mbody/`

and Anderson

`http://www.rpi.edu/~anderk5/lab/`

are recommended. Selected information related to some other software is reported in the following, and summarized in Table 1.

### 7.1 Chrono::Engine

The Chrono::Engine software is developed by Deltaknowledge, presumably a spin-off of “Università di Parma” (`http://www.deltaknowledge.com`). The library itself can be downloaded for free, but it can only be used for non-commercial purposes. A commercial license is available. The software can be redistributed, even after modifications, with few restrictions, but in general its license should be considered non-free (in the sense of freedom). All restrictions seem to focus on discriminating whether the final user will or will not make money out of its use. In the end, the user is not granted any freedom.

### 7.2 DYMORE

The DYMORE software is developed by the Georgia Institute of Technology, as reported on the web site `http://www.ae.gatech.edu/people/obauchau/`. Finally, in 2007, its source code has been made available through the above mentioned web site. Unfortunately, the software downloaded as of this writing does not compile; but the worst news is that it comes with no license! So there is no way to determine what freedoms the user has; as a consequence, it is advisable to avoid using or distributing it, neither as is nor modified. Note, however, that only version 2 is available for download. This version has been presumably obsoleted by version 3, which, as of this writing, is announced on the same web site, but is not accessible.

Table 1: Summary of software access levels and freedoms

Software	Access		Freedom				Developer
	0	1	0	1	2	3	
SIMPACK	—	—	—	—	—	—	INTEC GmbH
ADAMS	—	—	—	—	—	—	MSC/SOFTWARE
DynaFlexPro	—	—	—	—	—	—	MotionPro Inc.
NEWEUL	—	—	—	—	—	—	Universität Stuttgart
SAMCEF Mecano	—	—	—	—	—	—	Samtech
ROBOTRAN	—	—	—	—	—	—	Université Catholique de Louvain
SPACAR	✓	—	✓	—	—	—	University of Twente
MBDyn	✓	✓	✓	✓	✓	✓	Politecnico di Milano
Chrono::Engine	—	—	—	—	—	—	Deltaknowledge
DYMORE	?	?	?	?	?	?	Georgia Institute of Technology
EasyDyn	✓	✓	✓	✓	✓	✓	Faculté Polytechnique de Mons
POEMS	✓	✓	✓	✓	✓	✓	Rensselaer Polytechnic Institute

### 7.3 EasyDyn

The EasyDyn software is developed by the Faculté Polytechnique de Mons, and it is available at the web site <http://mecara.fpms.ac.be/EasyDyn/>. It consists in a library that can be used to generate the equations of motion of multibody systems, using the free tool MuPAD to perform automatic differentiation. The software is distributed under the GNU General Public License, and thus gives users all freedoms.

### 7.4 POEMS

The POEMS software is developed by the Rensselaer Polytechnic Institute. It represents an interesting effort to put together a computational environment for the parallel solution of large scale multibody problems. What is mostly relevant for the purpose of this discussion is that it was presented as a candidate for release under the GNU General Public License, thus making it free software. However, the project seems to be idling right now, and the software seems to be inaccessible.

### 7.5 Other Software

Listing all multibody software was not the intended purpose of this paper. The author apologizes with developers of any multibody and multibody-related software that has not been mentioned. Please feel free to contact the author, possibly providing any useful information, for a future, enriched version of this document.

## 8 Conclusions

This paper discussed what terminology is most appropriate when dealing with the access rights users and developers have to software, considering the specific case of multibody software. Examples of publicly known software are given, detailing their access level, and the potential implications of access levels and access rights on the scientific progress in applied multibody analysis.

A classification is proposed, based on the access rights users have on the software itself, as opposed to the conventionally accepted, but misleading, distinction between “commercial” vs. “academic”. In fact, all software that provides commercial grade support can be called “commercial”; the fact that access to the source code is granted or not has no relevance. What is relevant is the freedom granted to software users and developers. To be considered free, a software license must not restrict the use of the software, its modification and its distribution, but it must restrict modifications to the licensing terms, to preserve freedom during the distribution process.

## Acknowledgements

The author acknowledges the cooperation provided by those software developers (individuals and companies) that replied to the author’s requests in form of unrigged information on their products and activities.

## Notes

This paper has been already rejected by the journals “Multibody System Dynamics” and “Journal of Multi-Body Dynamics” before even getting reviewed, because considered out of the scope of the Journals by the respective Editors. It is the author’s opinion that its double rejection confirms the importance and the need for better awareness of the implications of software access in the field of Multibody Dynamics.

## References

- [1] Richard M. Stallman. *Free Software, Free Society: Selected Essays of Richard M. Stallman*. GNU Press, Boston, MA USA, 2002.