MULTIBODY DYNAMICS 2003
Jorge A.C. Ambrósio(Ed.)
IDMEC/IST, Lisbon, Portugal, July 1-4 2003

OPEN-SOURCE MULTIBODY ANALYSIS SOFTWARE

Pierangelo Masarati, Marco Morandini, Giuseppe Quaranta, and Paolo Mantegazza

Dipartimento di Ingegneria Aerospaziale, Politecnico di Milano,
via La Masa 34, 20156 Milano, Italy
email: {masarati,morandini,quaranta,mantegazza}@aero.polimi.it,
web page: http://www.aero.polimi.it

Keywords: Multibody Dynamics, Open Source Software, Cooperative Programming.

Abstract. *This paper discusses the opportunity, for the community of multibody developers and users, of the availability of Open Source multibody analysis software implementations. The basic characteristics of such implementations are drawn, and a critical review of what is available is presented.*

1 INTRODUCTION

This paper discusses the opportunity for the multibody community and in general for the organizations that are interested in the development of multibody analysis capabilities of having an Open Source[1] software for multibody/multidisciplinary analysis. The basic concept the Open Source paradigm relies on is very simple: if the source code of a software package is available to experienced and highly motivated users, those who have the knowledge to look at it, and really need a new feature, or want to get rid of a nasty problem, will eventually succeed, and make their improvements available to the community. Few decades of experience show that this utterly simple idea works, and its success is testified by the spread of pieces of software such those that make the internet the invaluable tool it is for everyday's work of millions of people, such as the Apache web server [2], the Sendmail MTA[3], the Linux OS[4] and thousands of small and large projects that cover most of the needs in different areas of everyday's computer usage. Openness doesn't mean total anarchy, however, and few rules are needed to allow the principle to work[5]. This is clearly stated, for instance, in the documentation that accompanies the General Public License[6] of the Free Software Foundation[7], which is one of the most significant efforts to protect the results of open software philosophy, essentially the source code, from intellectual property abuses while preserving its full availability in the broadest manner.

It should be made clear that openness must not be viewed as in conflict with proprietary software and software companies. As an example, companies like Sun, IBM, HP, and many others, have a very liberal attitude towards open source software, which means they use open source software to do business, and contribute to the community by releasing some of the software they produce; other companies fund organizations that develop open source software.

Looking at the technical and specifically at the engineering software field, some effort towards open source can be noticed, such as desktop computing environments like Octave[8] and Scilab[9], various data visualization packages; complete and reliable Finite Element Analysis (FEA) codes can be found (e.g. by browsing the Internet Finite Element Resources, IFER[10]); the code ASTER[11] should be cited for its completeness.

The idea of open source software in the community of mechanical analysis and significantly of multibody analysis seems to spread a bit more slowly than it does in other fields, possibly for various reasons.

There are fields somewhat related to multibody dynamics that are seeing a proliferation of open source initiatives. The object-oriented modeling language Modelica[12] is an open language specification for the modeling of multidisciplinary systems. Its strength is an open source model library, but its weakness is the lack of an open interpreter (the Open Source Modelica project is proceeding at a very slow pace[13]). The Open source RObot COntrol Software project (OROCOS[14]) is a noteworthy effort to design robot control software, focusing on software components for real-time, preserving as much abstraction as possible to be independent of any specific hardware architecture. Despite its ambitious goals, the project is proceeding at a very fast pace; recently, an effort to provide a high-level layer of kinematics and dynamics model description, of course geared to robotics and motion control issues, was started[15]. The Real-Time Application Interface (RTAI) project [16] represents an effort to provide a middleware layer, with POSIX compliant programming interface, for hard real-time programming on PCs running Linux. It eventually evolved in a complete real-time operative system, offering services like user-space real-time programming, and a plethora of real-time RPC services for network real-time programming. Hooks to popular simulation and control design environments like Mathworks's Realtime Workshop and Scilab, for automatic generation of real-time code, have been developed.

Most of these projects are partial examples of what every scientific community should struggle for, in order to spread knowledge and ease the research and development of new techniques without the need to restart every new project from scratch. Incidentally, they are somewhat related to the multibody field: Modelica's library contains a multibody analysis package that experienced a very fast growth, pushed by robotics, automotive, real-time and aerospace analysis needs. Orocos, on turn, for the part that concerns the control of manipulators, may benefit from multibody analysis for motion simulation, path prediction and more. RTAI is being used as part of the Orocos project, and in other open source multibody analysis tools, to provide reliable real-time capabilities that allow hardware-in-the-loop simulations and realistic predictive and look-ahead control capabilities.

Currently, to the Authors' knowledge, there is no analogous project in the multibody field, nor there is any significant effort to participate in these related projects to bias them towards the needs of multibody dynamics analysis. The only open source multibody packages currently available, as far as we know, are DynaFlex[17], DynaMechs[18] and MBDyn[19].

This paper draws a possible scenario for open multipurpose multibody analysis frameworks, discusses their requirements, and illustrates the benefits that the multibody analysis community: the academics, from both the research and the educational points of view, and the industry would gain from the availability of one (or more) open source software projects.

2 OPEN SOURCE MULTIBODY ANALYSIS SOFTWARE

This section discusses the opportunity of, and the requirements for, an open source multibody analysis software.

2.1 Requirements for an open source multibody software

Do requirements for an open source multibody software differ from those for a closed source one? The question is only apparently trivial. The main, phylosophical difference between an open and a closed source software is that the former is expected to undergo a distributed, cooperative development based on a loose, mostly voluntary-based coordination. As a consequence, an open source multibody software should emphasize properties like:

a) *modularity*, to clearly separate, and possibly hide, implementation details from abstract and general interfaces;

b) *versatility*, to allow different researchers and developers to implement their own ideas and approaches without affecting, constraining or negatively influencing the work of others;

c) *data abstraction*, to ease the evolution of the code without requiring major re-engineering; it should also be based on

d) *open language and development environment*, to ensure that no platform, language or development tool limitations affect its availability, accessibility and development process.

These requirements of course may apply to any piece of software; however, it is of utmost importance that they are satisfied in case of an open source project, because etherogeneous developers and users must be able to interact with loose coordination.

It is worth to notice that no requirements are set on the type of problems it should address. In fact, its openness and the above mentioned properties should allow researchers to implement the desired features within the existing framework, or push the framework beyond its current limits if needed. The existence of a community of experienced developers and users should guarantee that existing functionalities are preserved, or repaired if needed, and that new features naturally fit in a general development roadmap.

2.2 Reasons for an open source multibody software

Two questions should be answered before the discussion proceeds. Who would benefit from an open source multibody software? Who would be disturbed by it? When working at a new idea, regardless whether it is an integration scheme or a deformable element formulation, researchers usually do a draft implementation in some computing environment, such as Matlab, Octave or Scilab; then, if the idea seems sound, they may or may not proceed to an more robust implementation, e.g. to check performances on problems of realistic size, or to solve specific tasks. It happens very rarely that a complete software of nearly industrial strength results from such a development process, because a product of this complexity requires time, programming capabilities and a work organization that is typical of industry rather than academics.

Moreover, while the brilliant idea that results in a new integration scheme may be worth investing time and resources in its development and implementation, everything else a usable program needs, e.g. data handling, standard mathematical software incorporation, output handling and more, is not worth the effort because everything already exists and it is only a matter of putting together libraries and programs, glueing things together with some experience and programing skills. However, this never comes for free: this type of work is of little reward for researchers, but it is mandatory to implement anything more than a proof of concept.

The availability of a modular, versatile and general software would allow researchers to focus on integration schemes or deformable element formulations, to mention the previous examples, exploiting an existing framework for infrastructural services. These services would result from previous development, and benefit from long and accurate third-party review resulting from feedback provided by other developers and users.

2.3 Reasons for not having open source multibody software

This point is partially covered by answering the second question that was posed earlier, which is legitimate, although malicious. There are two categories that might be unhappy with open source multibody software: researchers that do not want to release capabilities, but only knowledge, and software producers.

In fact, academic research is useless unless it expands the knowledge of the community as a whole, without barriers. However to contribute to global knowledge is different from contributing to global capabilities. One could be tempted to preserve unique capabilities, e.g. the ability to do some specialistic multibody analysis, while still communicating the knowledge basis of one's work by usual publishing means. This position is legitimate, although questionable. However, it does not inhibit from participating and taking advantage of open source solutions; one can base one's specialistic problem solution on open source software and do not release it into the mainstream source. Most open source licenses allow this, possibly with the constraint that the resulting closed software cannot be distributed in binary form only.

The second category of potential open source opposers is mentioned only to underline that, although an open source multibody software designed and developed under the previously mentioned guidelines would closely approach some of the features of commercial software, it is not necessarily and immediately in contrast or in competition with current commercial products. The basic aim of an open source software is to produce a framework for multibody research which allows experimental software to quickly and easily reach a readiness level that makes it of practical use for actual analysis and not only for proof-of-concept applications. The extra value that commercial software will ever offer is essentially support and (limited) liability.

2.4 How it works

After agreeing on the reasons that justify the need for Open Source multibody analysis software, and after drawing its essential features, a look at success stories of Open Source software shows that some ingredients are mandatory.

i) *A working startup implementation*: experience shows that, with due exceptions, projects starting from scratch have little chance of gaining acceptance, because it is obvious that only a usable software, even if under limitations, can have a community of users and potential developers. The need to improve what is already in use will make the users turn into developers.

ii) *A suitable license*: it is also shown by experience that licensing issues are important, and should not be ignored. A loose license may allow malicious software producers to hijack rights on open software, while too tight a license may prevent researchers from cooperating. It is the Authors' opinion, although not supported by any competence in the legal field, that a reasonable trade-off is represented by a license that allows the broader use of the software, but prevents its closure to any extent. As a consequence, everybody should be allowed to make any use of it (i.e. the community of users and developers can expand without restrictions), including undisclosed development for personal or internal use, but changes and improvements must be released in open form for any purpose (i.e. the community will benefit of improvements), under the same license terms of the original software.

iii) *A reasonable documentation project*: users need training to gain confidence in software. However, software training, and even basic software documentation requires resources that are usually not accessible by academic and research institutions. Of course, the documentation and the support does not have to be comparable to that of commercial software. In this statement, what "reasonable" means must be clearly defined, by deciding what commitment the members of the community should put in documenting the software and its usage to reach an acceptable level. Here again experience may support us: usually, skilled developers do not need code documentation; all they need is to read the code, and, in this specific field, the literature on the subject for problem-related issues, so software documentation may be a secondary problem. On the users' side, usually the documentation is written, or at least widely improved, by enthusiast users, in form of HOW-TOs (or HOW-I-DID-ITs), after they succeed in making it work.

3 A CASE STUDY: MBDYN

One might ask: why this discussion when MBDyn is already available? As was previously mentioned, MBDyn is a multibody multidisciplinary analysis software that is biased towards aerospace engineering, while preserving a general applicability in the mechanical engineering, robotics and control fields. References to some of the works performed with MBDyn can be found on its web site[19]; they are not mentioned here because this document is focused on the software development and not on its use.

This section will discuss why MBDyn, although being only a partial answer, at least in its present form, to the need for general and generic multibody software, can be a valid basis for an open source multibody analisys framework.

3.1 Brief history

As happened to many valid open source software that are currently in use, MBDyn started as a proof-of-concept code, which eventually became general purpose and lately was released as open source under GPL 2[6].

As a consequence, it is limited by some of the assumptions the initial proof-of-concept code was based on, i.e.:

- it is biased towards direct dynamics and Initial Value Problems (IVP);
- it lacks a friendly Graphical User Interface (GUI);
- it does not support event-driven solution (hybrid solution) yet;
- it is not consistently thread-safe, thus preventing some potential evolution directions;
- its data structure, in some portions, suffers from limited and unevenly distributed modularity and abstraction.

The program is essentially coded in C++, which currently represents the best trade-off between object-orientation and performance. In its initial implementation, in the early 90's, it was coded in F77; its career as a proof-of-concept would have been very short because of unbearable maintenance problems. The need to investigate the use of a multibody approach to the analysis of helicopter rotor active control, a truly multidisciplinary problem, required to reimplement it in a more flexible language, with different layers of abstraction. As a consequence, it proved to be flexible and general enough to allow its enhancement in many portions of code. It is actively maintained and developed, and it is growing in different directions, as will be explained later.

3.2 Current Status

Eventually the developers, pushed by the need to overcome some of the above mentioned limitations, re-engineered the software to make it more abstract, general and versatile. Examples of successful improvements were the generalization and abstraction of the linear solution portion of the software, which allowed to add three radically different linear solvers with very limited and localized code specialization, or the basically straightforward coarse-scale parallelization of the solution, or a very recent improvement that allows to use radically different nonlinear solution strategies (direct vs. matrix-free solvers) without affecting the problem description part of the software.

Some of the F77 parts were maintained and incorporated; parts of the early linear solvers and some well proven aerodynamics routines are still present, although most of the software is now in C++ and most of the standard utilities (linear solvers, network communication, runtime linking, special purpose software) are delegated to widely available libraries.

Some portions of code, which are not strictly specific to the multibody analysis implementation, have been coded in C, to allow an easier reuse without the sometimes scaring programming overhead of C++.

The team of developers is very compact, so there have been only very limited critical cooperation phases. Development strategies are loosely coordinated among the stable developers. Some of the analysis modules (e.g. the flexible element, the hydraulics library, the parallelization of the solution, and the real-time simulation support via RTAI) have been concurrently implemented with the cooperation of undergraduate and Ph.D. students, and a good task separation was easily obtained by coordinating critical changes and by enforcing an appropriate use of concurrent development tools (e.g. Concurrent Versions System, CVS[20], but more sophisticated tools are available).

No code branching and no separate development trees have ever been required; despite the number of etherogeneous concurrent developers, the software always evolved as a whole.

3.3 Lessons learned

The problem-oriented software evolution, with limited look-ahead to foresee, and account for, possible growth directions, allowed the developers to experience the need for more abstraction and generalization, and to cast the lesson they learned in the notes presented in this paper.

In this sense, the approach followed by other projects, like OROCOS, is much more structured, with a solid initial analysis phase. This is mandatory since the project is really distributed, so coordination is fundamental. Moreover, OROCOS is EU funded, so all the design and development phases must be carefully documented, and bureaucratic issues probably are of much concern.

On the contrary, MBDyn is currently self-funded, developed on a voluntary basis, and its evolution is mostly driven by the need to fulfil special requirements by committers of research contracts, apart from internal research objectives.

MBDyn only partially meets the criteria exposed in this paper, but it is a good starting point to develop an open source multibody simulation framework because it is already available, it allows to perform a wide spectrum of multidisciplinary analyses, and it is actively developed and maintained.

A possible limitation, which by the way is common to most academic softwares, is the lack of a friendly GUI. Unfortunately, its development may require large resources and, owing to the absence of good open source high-level graphical packages, it might require a dedicated project. In fact, the required knowledge, skills and resources are mostly unrelated to those of typical academic institutions involved in multibody research. Furthermore, the needs of software for multibody applications could be fulfilled by GUI software for broader applications. The reuse of existing visualization packages is being partially explored by developing postprocessing tools for OpenDX[21], an open source package for visualization of scientific data (formerly IBM's Visualization Data Explorer), and partial pre/post processing interfaces for commercial multibody analysis softwares (MSC's ADAMS/View[22], formerly MDI's, and Altair's MotionView[23]). As an example, the rendering of the semispan tiltrotor model of the V-22 is shown in Figure 1.

In any case, it is clear that a larger open source multibody project would require an informal structure, with voluntary developers and users free to work on the preferred topics but within a given framework, at least for those parts that would be integrated into the mainstream code. High-level decisions should be based upon consensus between active developers, and a steering committee might be advisable, to speed up the decision process and provide a clear view of the development streamlines. This should be achieved in a mostly informal way, to allow everybody to express their creativity, and to preserve researchers' freedom.

4 CONCLUSIONS

The need for Open Source software frameworks in the multibody research field has been discussed, and possible evolution strategies have been depicted. Currently, the only real operative Open Source solution available is MBDyn, which partially meets the requirements for a common multibody, multidisciplinary analysis framework. Based on different Open Source software development and use, MBDyn is proposed as a starting point to de-

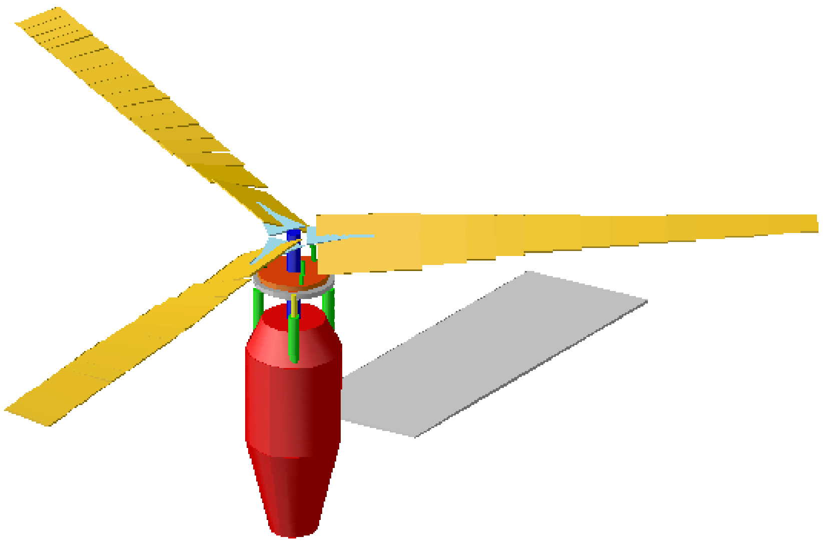
Figure 1: Semispan model of the V-22 tiltrotor (rendered with ADAMS/View)

velop a multibody simulation framework, to form a community of Open Source multibody software users and developers.

REFRENCES

[1] Open Source Initiative (OSI); `http://www.opensource.org/`

[2] Apache; `http://www.apache.org/`

[3] Sendmail; `http://www.sendmail.org/`

[4] Linux; `http://www.linux.org/`

[5] Eric S. Raymond; "The Cathedral and the Bazaar"; 1998; `http://www.openresources.com/documents/cathedral-bazaar/`

[6] General Public License; `http://www.gnu.org/licenses/licenses.html`

[7] GNU/Free Software Foundation (FSF); `http://www.gnu.org/`

[8] Octave; `http://www.octave.org/`

[9] Scilab; `http://www.scilab.org/`

[10] Internet Finite Element Resources (IFER); `http://www.engr.usask.ca/~macphed/finite/fe_resources/\ fe_resources.html/`

[11] EDF's Aster `http://www.code-aster.org/`

[12] Modelica; `http://www.modelica.org/`

[13] Open Source Modelica; `http://www.ida.liu.se/labs/pelab/modelica/`

[14] Open source RObot COntrol Software (OROCOS); `http://www.orocos.org/`

[15] OROCOS "Kinematics and Dynamics" document draft `http://www.orocos.org/kindyn.html`

[16] Real-Time Application Interface (RTAI); `http://www.rtai.org/`

[17] DynaFlex; `http://real.uwaterloo.ca/~dynaflex/`

[18] DynaMechs; `http://dynamechs.sourceforge.net/`

[19] MultiBody Dynamics (MBDyn); `http://www.mbdyn.org/`

[20] Concurrent Versions System (CVS); `http://www.cvshome.org/`

[21] OpenDX; `http://www.opedx.org/`

[22] MSC.ADAMS; `http://www.mscsoftware.com/`

[23] MotionView; `http://www.altair.com/`