Transparency and control in engineering integrated assessment models*

Minh Ha-Duong†

25th May 2001

Abstract

Better software engineering such as archiving releases with version control, writing portable code, publishing documentation and results closely tied to the code improves integrated assessment models' transparency and control. A case study of four climate change policy analysis models found that source code and data was generally available, but for larger models licenses were more restrictive with respect to modification and redistribution. It is suggested that Free software licenses such as the GNU GPL would improve transparency and control. Moreover, opening the source allows opening the development process, a potentially important tool to improve collaboration, data sharing and models integration.

1 Introduction

Discussing methodological challenges for integrated assessment models, Rotmans [11] states that the evidence for integrity is transparency. Following Ravetz's suggestions in [9], he stresses the need to formulate some guidelines for good practice.

Building upon these previous works, this paper formulates guidelines to improve transparency and control in integrated assessment models. The originality is the software engineering point of view.

Schneider discusses in [12] the vital challenge for the profession to communicate transparently about the models' limits. This is all the more delicate since models apply to hot environmental issues. While social and political aspects of model use are important, this paper will not deal with this type of external credibility questions. The focus is on scientific transparency and control from within the community of peers.

Even in this restricted sense, transparency has several meanings. It can be understood as the fact that all the variables, parameters and equations in a model are well understood theoretically. Another meaning of transparency is that there is a simple and direct correspondence between the model and the reality. These two goals are desirable

*$Id: transp.lyx,v 1.26 2001/05/25 14:45:37 haduong Exp $

†Chargé de Recherche au CIRED-CNRS, Centre International de Recherche sur l'Environnement et le Développement. Campus international du Jardin Tropical, 45bis avenue de la Belle Gabrielle, F94736 Nogent sur Marne CEDEX, France. Correspondence to haduong@centre-cired.fr.

but conflicting, since increasing the size of the model to satisfy the latter reduces the theoretical control.

Another aspect of this trade off between intelligibility and detail arises when one follows Morgan and Dowlatabadi [7] guidelines and sets the central focus on uncertainty analysis. Recognizing the multiplicity of future states of the world certainly helps to capture key social dynamics related to anticipations and risk reduction attitudes. At the same time, the need to use advanced mathematics to represent the relevant aspects of ignorance theory may undermine the credibility and the understandability of results.

Control also has different meanings. Some issues are related to numerical analysis, for example the quality of optimization, integration and differentiation algorithms. A second aspect of models is the stability of results and their robustness with respect to small changes in parameters. Third, moral and legal controls restrict who can access, disseminate and interpret the models and how. Through anonymous reviewing and other institutions, there is an important peer control system over the scientific aspect of modeling activities. How does this peer control system performs with respect to engineering integrated assessment models? Is vastly improving transparency necessary and sufficient to improve quality?

Section 2 discusses in more detail model transparency issues, in general and in a case study of four climate policy models. This case study lays the groundwork for the second part of this paper inspired by the contemporary free/open-source approach to software licensing. Section 3 will discuss the question of open source, that is releasing code after publication of results, and the question of open development, that is releasing code before publication. Section 4 concludes by summarizing findings and discussing practical implications.

2 A case study with climate-energy policy models

2.1 The problem with models: quality control

Control in science is done through many formal and informal institutions such as grant selection comitees, awards, direct group interactions and so on. One important institution is the peer-reviewing system: when an author sends a manuscript to a scientific journal, the editor asks independent and anonymous reviewers to give their advice on the suitability of the work for publication. The editor and referees are peer researchers.

The referees' precise mandate varies across scientific communities. For example the editor of a mathematical journal could mandate the referees to check proofs step-by-step, whereas an editor in economics could ask them to trust the authors on mathematical details.

When it comes to model-based results, verifications could be led at different levels. One could run the model again with similar hardware and software environment. One could run the same program, but on a different machine or/and under a different operating system. Ultimately, one could start from the model equations and re-code it from scratch in a different programming language, thus gaining the most intimate knowledge of the model.

One problem with integrated assessment is that referees are usually not asked to verify the model-based results of submitted manuscripts, not even audit the source code. They only assess papers against existing literature, their intuition and understanding of the field. This reflects the judgment that, while the costs of formally reproducing the results are presumed large, the additional benefits are presumed small.

This is not unique to integrated assessment modeling. It occurs in many experimental disciplines. But for scientific experiments, even if no reproduction is done *ex ante*, insuring reproducibility is a fundamental design goal. Yet contrary to chemistry or optics, integrated assessment papers usually do not contain enough details for a different team to reproduce the published results.

To understand why reproduction is costly in terms of useless duplicated work and publication delays, one has to consider that integrated assessment scientific and economic models are specialized software products usually built in an academic research context. They are not designed for repetitive, everyday use, but for unique though experiments. They are often in a state of flux, because they need to evolve with the world they describe, its scientific understandings and with the interests of stake-holders financing them. All this explains why it would be a waste of resources and time to code these research tools with the same stability and robustness levels as, for example, commercial aircrafts control systems. It is not a critical failure when a model crashes because the population number in the input datafile is negative.

Having only an informal check of the models is unsatisfactory for several reasons. First, there is a risk of pure and simple error. Correctness of the model is often central to the arguments being made. Yet correcting a small bug can often lead to significant change in the results. Second, the quality and maturity of code underlying a model can be highly variable. As long as code escapes peer review, there are no incentives to reward the efforts involved in writing better code. Third, it is a matter of principle that the deeper the independent review, the better.

Reproducibility is all the more important for virtual thought experiments have an enormous legitimacy problem: models do not have the formal strength of mathematical proofs, nor the legitimacy of real experiments. Little remains of integrated assessment if the protocol, in fact the computer program, cannot be run again with the same results.

It is essential that published results are objective, that is independent of who runs the model. Therefore, one cannot but ask for external reproducibility as part of an ideal standard of scientific transparency and control. Yet reproducibility is not an all-or-nothing concept. As said above, verification of model based results can be led at different levels. This is also a matter of how much means and know-how are available.

In order to better grasp how to improve transparency, we need to address another issue, that reproducibility is only a potential concept: it refers to what could be achieved eventually, given reasonable means. This can be subject to very different subjective interpretations. Let us outline four objective characteristics of models that imply better transparency.

Internal reproducibility: Maintaining internal reproducibility requires that the source code, complete with the data that was used to produce the results and enough notes on the procedure followed should be archived. In order to do this, there is a need to manage closely the evolution of the source code. A version control

system is useful to achieve this. Moreover, running the model should not depend upon a single critical person in the research team.

Portability: A program is portable when it can be use on a different computer with little or no modifications. For example, GAMS models are very portable, but DOS batch files are not. Explicit consideration to this opens the possibility for reproducibility and external reviewing of the model itself. Writing a portable program requires much work, but the benefits are huge: it usually contains less bugs and greatly extends the usefulness of the model.

Automation: One sure way to achieve consistency between the model and published results is to do all the computations necessary for producing the paper's tables and figures automatically. This means having scripts that move along the bits all the way from the input datasets into the model, and then feed the results into the typesetting and plotting programs. Writing this kind of scripts is simple compared to writing the substance of models.

Integrated documentation: The most practical way to achieve up to date documentation is to integrate documentation and code in the same file. This is known as literary programming. Most modern programming environment offer facilities for this. For example, *Mathematica* allows to write notebooks containing at the same time active equations and formatted text. For Perl, C, C++ or Fortran, literary programming is done by including documentation within special comments around the code. A simple tool extracts this documentation and typeset it into human-readable technical manuals.

The model having several of the features outlined below will be more reproducible in the sense that the cost of reproducing it decreases. Note that transparency is not a measure of model usefulness. Whether others will actually reproduce the model does not only depends upon the cost to do so, but also upon the benefits, that is, as R. Tol remarks (pers. comm.), if the model has a better problem definition than competing models.

Experience also warns that there is about an order of magnitude of cost increase between a program designed to run once on a given system, and a software system designed to be re-used by third parties in the future.

With respect to the four features above, it may seem that large-sized models cannot afford as much transparency as small models. On the contrary, larger models do need more transparency in order to be understandable and manageable. The illusion that a model will be better documented later is dangerous. The idea that a small hack, that is code quickly written and poorly documented, could ever evolve into a large model without putting in transparency first is very risky.

There is not an absolute best transparency goal, but only appropriate levels according to the ultimate model's objective. Authors strive for the different levels of reproducibility according to their different publication goals. Manuscript notes in the lab notebook may insure internal reproductibility and be satisfying for working papers and research reports. Automation may suit some situations such as PhD thesis work or

class-A journal articles. Portable models can been found in many books such as [5], [8].

This discussion hints that more articles founded on version-controlled, automated, portable and literary programmed integrated assessment models would greatly improve transparency and control in the field. To some extend incentives are needed, since they require significantly much more discipline from the authors, but these software engineering practice are also often worthwhile in themselves to the authors. At the very least, editors should require the authors to disclose clearly and publicly the objective reproducibility features of their model results.

Admittedly, there are many other aspects to transparency and control, and these are only a few ideas aiming to tilt the playing-field in the right direction. Improvements are needed because at the present time one may question whether most of the published works are even internally reproducible. How many IA models are under version control? How many authors can easily reproduce their previously published results? These questions should not be answered without serious empirical investigation. What we will see next is that the highest reproducibility levels are not an unrealistic goal and have indeed been achieved by several models.

2.2 Illustration on climate change integrated assessment models

Models discussed below are influential in the field of energy policy and climate change analysis arena. These are MERGE [5], IMAGE [1], DICE [8] and MARKAL [3]. This is a convenience sample illustrating the diversity of possible model sizes. Each model, at its scale, can provide some type of informations about what happens to the climate change issue if different energy policies are chosen.

Versions presented here are all outdated. This illustrates that these models are under active development. However, obsolescence will not be an issue here because the point is not to judge these particular models. They are only used to illustrate general limitations and potential areas of improvement in the field.

The top half of Table 1 summarizes features of the four models. All of these models are distributed as source code using the media indicated. They require a compiler which, except for IMAGE 2.0, is the GAMS programming language, with an optimization solver such as MINOS5 or CONOPT. Above all, putting these models to good use requires significant human expertise.

Models in the table are arranged by increasing size. The larger the model, the more it is a team effort. In order to give an idea of the nature of these models, the table shows a “Difficulty” line. This line reflects the author’s personal judgment about what constitutes the less transparent and hardest to control area for each particular model. This mostly illustrate the diversity of issues involved in trying to improve on those grounds.

The focus now switches away from the models themselves onto their license agreements. Studying the license agreements is relevant in order to discuss transparency and control because these texts are short, important, and comparable across all models. License agreement is one of the few documents where the interests of all the parties involved in scientific and economic models need to meet. Scientists need to publish in learned journals. The modeling team can be expected to have a large emotional as well

Model	DICE 94	MERGE 2	ANSWER MARKAL 3	IMAGE 2.0
Owner class	Professor	Institute	Global Consortium	National Institute
Owner	W.D. Nordhaus	EPRI	ETSAP	RIVM
Size	200 lines	2000 lines	Matrix file is 1Mb	40 000 lines
Difficulty	Economic theory	Nonlinear Opt.	Get own data [1]	Computing technicalities
Model media	Web, book	Web, book	CD	Results and code CDs
Allowed uses	Any	No Competition [2a]	Any	No Competition [2b]
Publish results?	Yes	Yes	Yes	With permission
Give feedback?	May	Must [3]	Must within 30 days	Must [4]
May modify?	Yes	Unspecified	Yes with approval	Yes
Redistribute?	Yes	No	No	No

Table 1: Characteristics and license schemes for four integrated assessment models. Source: Agreements signed with respective model owners. Notes: [1]At date of writing, a common international technology database is in preparation. [2a]Can use the Proprietary Information to analyze carbon dioxide abatement costs for countries other than USA. [2b]Can not use the model for the benefit of third parties, for Dutch or European policy applications or for international organizations. Can use the model for domestic national policy analysis, provided RIVM has full access to it. [3]Must report applications which are in the public domain, and a letter report documenting the overall experience in applying the model, including suggestions to improve portability and user-friendliness. [4]User must report problems and make modifications available for RIVM use.

as intellectual investment in the model. The sponsoring institution may be interested to sell something. The stake-holders want independent expertise such as nationally established databases and models, they are interested in official quality results based upon well-established practices.

This is not the place to reproduce in full or in part the text of licenses agreements themselves, so Table 1 summarizes. To read the bottom half of Table 1, it is fundamental to have in mind that intellectual property rights can be finely subdivided. For example, buying a videotape at the supermarket carries the right to play it at home, but not to broadcast it on a public network. In this respect, intellectual property is different from most other consumer goods or with real estate, where ownership is usually associated with full property rights. Each line represents a particular usage right that the license does or does not grant. As shown, all licenses provide the right to use the model and publish results with only minor restrictions.

But beyond this, differences are large. Most may be explained by considering the ultimate goal of the license:

- W.D. Nordhaus states that the DICE 94 code is "made freely available to users and is not proprietary". He explicitly uses for software the license scheme implicitly used for scientific ideas: everyone can use a published work or mathematical formula. DICE has been extensively re-used and modified by many third party researchers.

- On the paper, the MERGE license is more restrictive than DICE's. In fact, the wording of this license seems caricatural of the silicon valley Non-Disclosure Agreements (NDA). However, the apparent rigor of this NDA is compensated by its narrow scope: it applies only to model-related information "not generally known to the public". Since MERGE is described in a book ; most of the code and data is available from the authors' web site ; and its updates are regularly published, it can be argued that most of it is publicly known. As a result, for all practical rights except redistribution MERGE is not restricted. It has been widely re-used externally for both academic research and policy analysis.

- The ANSWER MARKAL 3 license is the only one with "payment" in it. A bare academic version might be obtained at no cost. But since using this model (as any other) requires significant financial investment in hardware, compiler, solvers, and human training, it makes sense to pay for a graphical interface shell (MUSS or ANSWER). The funds for MARKAL go to an international consortium (ETSAP) hosted by the International Energy Agency which offers technical support and other services to the model-using community. The main purpose of the license seems to be protecting the viability of the consortium. As with the above two models, dozens of teams have used MARKAL and many different versions have been contributed

- Finally, the IMAGE 2.0 license is the least permissive. I contend here that the main barrier to IMAGE transparency and control is the practical complexity involved with independently running the model. There are few running versions of the model outside RIVM, most of them implemented by the original team.

IMAGE 2.1 has been independently run at the Laboratoire de Météorologie Dynamique (Université de Paris VI), in collaboration with CIRED and RIVM. Originally compiled on HP-UX machines using some Fortran extensions, the model received modifications to be ported on Appolo, Sun or Linux machines, with standard and more strict compilers. Results have not been integrally reproduced yet.

This review shows that the license under which integrated assessment models are distributed generally, except for DICE, do set significant limits to their use and redistribution. This has not precluded successful academic work, but one could wonder to which extent it slowed the general progress in the field.

This comparison also suggests that presently, the bigger the model, the more closed is its license. This can be understood in a competitive business logic, where models are regarded as production tools, and licenses are here to protect the intellectual investment. This is in conflict with the common academic research logic regarding models as scientific tools.

Conceptually, transparency and peer control would greatly improve if all models followed the DICE license scheme. But are such open licenses practical for large models? The sequel of this paper discusses under which situations large software projects can thrive under the most open license schemes.

3 Ideas from Free/Open source software

3.1 Ultimate transparency ?

The highest standards of software transparency and control are defined in license schemes used by the free software/open source movement. In particular, the most convenient and widespread way to guarantee the ultimate transparency of source code is to use Richard Stallman's General Public License, also called the GNU GPL. Part of this seminal document on transparency and control is reproduced in the annex.

The GPL was written in 1992 as the cornerstone of an effort to build a Free clone of the Unix operating system. This effort is known as the GNU project, the acronym standing recursively for "GNU is Not Unix". In short, the GPL states that the program code can always be used, modified, compiled and redistributed by anyone. Moreover, it states that if modifications to the code are distributed, either free or for a fee, then they must also be under the GPL.

In that last clause, it is more radical than the DICE license, which does not impose any further obligations to anyone re-using the code. The question of whether or not to impose obligations to re-used code is a controversial issue. But that controversy is not fundamental with respect to transparency and control of integrated assessment models. In the sequel of this paper, a free/open source model will mean either a model under the open academic DICE-style license, or a model under the Free Software-style GPL.

Note that in this expression, the word Free does not refer to the price of software, but stands for not restricted. Another way to make this point is to note that the appropriate French translation is *libre*, not *gratuit*. Colloquially, this is also referred to as the difference between Free (as in Speech) and free (as in beer).

Three out of the four integrated assessment models examined above were not licensed as free/open source. One reason for this is that the GPL is relatively new: the first versions of these models discussed were written well before 1992. That is why this paper aims to increase the general awareness about its utility. Wide use of open source licenses would be a low-cost but significant improvement in transparency and control.

Some factors governing the choice of the license model are beyond modelers control: The public or private nature of funding is important. Parts of the model and some data from external sources may carry their own license. And the institution where the model is developed usually owns the intellectual property and may have a pre-defined licensing policy.

In the latter case it must be clearly conveyed to the bureaucracy that the quality of IAM depends essentially in its transparency. In order to facilitate that communication, the modelers need to grasp the implications for themselves of a free/open source model.

It cannot be under-emphasized that sharing the code of the model is absolutely separated from giving the right to use the name of the model. Using the name is an issue of trade mark, whereas using the code is an issue of laws protecting inventions. The soft drink market illustrates the situation where the product can be copied, but firms defend their brand name as their main asset. For de-materialized goods such as models, this separation between substance and image may not be as easy to understand and implement. One of the reasons for this is the heterogeneity of legal culture around the world.

Consequently, each time a model's code is made public, the author must remind the readers that the model's name remains the strict property of the owners. While the code is public, it remains that the name DICE cannot be used to designate anything else than the original DICE model. The use of the (TM) sign is a common business practice to acknowledge ownership every time a trade mark is used, but not a legal obligation in most countries. The common and accepted scientific practice is not to use that sign, but simply to cite the author.

It may be to the owners interest to grant liberally the right to use the name and fame of their model. Doing so helps to disseminate it widely and ultimately lead to a better understanding by the community of peers. But this is an indirect effect. At heart, transparency and peer control regard the models actual code, not reputation. To open the source code, it is not necessary to weaken controls upon the model name in any way.

One problem that could arise from public release of the code and data is that it implies no ex-post control of the finished product. More specifically, publicly available model code can be used to find politically different results that displease the original team or sponsoring institution. I think that this risk should be discounted on the ground that criticism is a normal, desirable part of science. Constructive use of models, if used correctly and not purely as advocacy tools, lead to interesting research questions about the models and the real problem at hand.

In summary, I believe it will be consensual that once results are published in a peer-reviewed journal, the source code and data cannot be kept secret but should be open for examination as a matter of principle. There are many practical ways to achieve this. The most straightforward way to achieve transparency is to place all files under the

GPL and archive them to a public, anonymously accessible repository online.

3.2 Control and collaboration

After having discussed the release of published models, we discuss the idea of opening the source code even before the work is finished. This takes transparency and control to a higher degree. Opening the source before publication is important for two reasons.

First, when integrated assessment models become complicated, understanding the code is impossible. At that point, the model is often criticized as a useless arbitrary black-box. Improving transparency and control in the process of model-making is a way to avoid this pitfall.

The second reason has to do with software engineering and regards Free/open source as a key to a more collaborative way of writing models. The central idea is that releasing publicly the code of projects induces other people to use the program, and these users will report useful feedback on how the code should be improved. When the coordinator releases source code early and often, it allows detection of errors by peer-reviewing at an early stage of the product life-cycle, when they are cheaper to correct. Eric S. Raymond [10] used the term "bazaar" to describe this development method, as opposed to "cathedral" building which describe well-ordered and pre-planned development method.

Interestingly, this transparency does not come at the cost of losing control over model development. It may be feared that, when anyone is encouraged to modify the code, this may lead to a profusion of different, incompatible versions and inefficiencies due to duplication of work. Experience shows that this variant of the tragedy of the unmanaged commons does not happen even for very large, complex software projects. The control of Linux, for example, clearly belongs to Linus Torvalds.

Various theoretical reasons have been set forward, see for example Lerner and Tirole [4], Moen [6] or Raymond [10], to explain why people contribute and why project leaders usually keep control of open-source programs development. This is not the place to summarize that literature, but to note that in practice it seems that people with the skills to make modifications to open source code understand the value of preserving design integrity in software architecture.

Of course, control is not guaranteed forever. If the project leaders are doing a bad job, a competing team can start developing the same code base in a different direction. This is called "forking" the project. But the contrary also happens: parallel projects will merge and pool their resources much more easily if both are under a free/open license.

On top of that, from a management point of view, openness of source code is a key element for control by sponsoring institutions and peers. It's always true that control requires transparency, for example governments require firms to publish their accounts. This is especially true about programming, where lack of communication and a proprietary behavior about code is a recipe for failure.

Still, the fact remains that making unfinished models externally available is even less common than making the published ones available. Two sets of reasons may help to explain this fact: concerns for potential mis-use beyond those discussed previously, and doubts about the efficiency of an open collaborative development method.

The problem with the open development process is that the risk of a competing team publishing sooner cannot be eliminated. Indeed, it exists even when presenting informal ideas at a scientific workshop. In some cases, it is unclear to what extent this concern is legitimate. For example, some publicly-funded atmospheric composition data is made available immediately, that is, as soon as the plane lands even before the collecting team analyzes it.

And paradoxically, with respect to this concern, publicly releasing early work may be better than trying to keep it secret, because releasing is a way to positively assert authorship and anteriority. A necessary condition for this, however, is the existence of a public archival system trusted to authenticate the authorship and the date of model releases. To our knowledge such a place does not exists today for integrated assessment models.

A second serious concern is that preliminary results could leak to the media before peer-reviewing validation. Scientists genuinely change opinions with new findings, whereas mass-media have different motivations and time schedules. I believe that preliminary results could be distributed under restricted diffusion without hurting transparency too much, since the validity of models should be assessed primarily on their structure and on their input data, rather than on their results.

Turning to doubts about efficiency, the objection is that opening the source of integrated assessments models at an early stage before publication would not help in ultimately developing a better code.

This objection may be based upon the idea that the potential pool of contributors may be too small. Indeed there are not many integrated assessment experts to constructively comment on the overall structure of the model. But there are many disciplinary experts that could provide useful insight about the model within their field. In fact, to show that the expected contribution from users is non negligible, we notice that in all four models discussed above the authors explicitly ask for feedback. According to our personal communications, they received some and found it useful.

That said, these models are successful and widely known. Not all new open-source projects succeed in attracting a critical mass of users to sustain interest in themselves. With respect to this risk, integrated assessment models are not different. However, it does not detract much effort from the core programming team to make ongoing work available. After all, once a file version control system is in place, it's only a matter of allowing external read-only access.

Regularly releasing ongoing work is also beneficial from a software construction control point of view. In large projects spanning several months, it is useful to have checkpoints at which the model is running, even if it does not has all the bells and whistles yet.

Whether it is a good idea or not to open the source before publication also depends upon the particular kind of model being built. Raymond [10] identified a set of criteria that makes opening the source more attractive.

1. Opening the source is a necessary condition for independent peer review. This criteria is very compelling in the context of integrated assessment models, since for most of them, independent peer reviewing is necessary to support correctness of design and implementation.

2. According to Raymond, software engineering history shows that peer review (therefore open source) is the only scalable method of achieving high reliability and quality. Reliability is not crucially important for some integrated assessment models which are use-once software. But the situation would be different for a meteorological model run everyday, or for an integrated model so large it had to run for months, as some climate models do. And while scalability may not have been a primary design goal for models in the past, recent development of DICE, MARKAL and MERGE have relied upon running in parallel one version of the model for each world region.

3. When software is critical to the user's control of his/her business, the user does not want to be locked-in with an outside supplier. To explain this intuition, consider for example that energy policy tends to rely a lot on models. In this situation, one can expect analysts and policymakers to ask for public models if they want to encourage a variety of opinions, instead of receiving a small number of specific modeling teams point of views. This may be one of the reasons explaining the recent interest in MARKAL by the US EPA.

4. When key methods (or the functional equivalent of these) are part of common engineering knowledge, there is not much to lose by opening the code. Some models may be structurally innovative. For example, the ICLIPS model [13] is governed by differential inclusions, which belong to the mathematically advanced field of non-smooth analysis [2]. This is not to suggest that ICLIPS is or should be less open than other models, but to illustrate a situation in which the core algorithms used may have a significant business value. On the other hand, many models use well-known key methods, as it is much easier to communicate about the results. MARKAL for example relies on linear programming.

5. Open source is useful to propagate open standards. Consequently, it is used for software that establishes or enables a common computing framework. The primary purpose of most integrated assessment models is not to build communication infrastructure for the Internet. But models can be seen as data-processing boxes using common data repositories. Not only do they need present and past observations, they also input elements of future scenarios such as those produced by the IPCC or the WEC.
   Some of these repositories have kept data behind a proprietary, closed interface such as an MS-Access database. This clearly goes against providing a transparent standard to make data generally accessible. In any disciplinary field it is useful to have common standards to exchange data to facilitate inter model comparisons and data re-use. This is all the more important in integrated assessment where managing the input and output data is an important and time-consuming part of modeling work.

A complicating factor is that integrated assessment models, by nature, tend to integrate different modules. Actually, it may be a common situation that parts of a model rely on re-using existing code or data for which the modeling team can not change the license. In this case, not all modules need to be ruled by the same license. While the

compatibility between license across modules is a tricky question, there are *a priori* no reason to go for the lowest denominator.

In particular, within a given model, the data files are often worth to open. Also, the code that realize integration between different sub-models may be a good candidate to early public release. This is because it needs review from other researchers, it needs high reliability and scalability, it does not involve commercially valuable algorithms and it establish a common computing framework. The latter criteria means that publishing the coupling code allows other researchers to understand the interface and eventually replace, or plug-in, a sub-model with another.

While not all integrated assessment projects would find an interest in open source modeling, the criteria discussed in this section may suggest situations in which the payoffs of managing an ongoing software project as open source will be large.

4 Concluding remarks

4.1 Summary of findings

Engineers, experts and scientists involved in integrated assessment modeling have established a small community with its journals and scientific society. This paper discussed how modern software engineering practices would remove some unnecessary barriers against transparency and control within the community.

Transparency and control are fuzzy and ambiguous notions. Reproductibility is a must for scientific experiments, yet it is rarely done for scientific and economic models, because they are evolving and one-of-a kind software products. This is the fundamental problem since these models do not have the legitimacy of real-world experiments.

Reproductibility, that is the cost of reproducing results, is an approximate measure of transparency. These four model features help reproducibility: 1/ Using version control tools[1] allow to archive the source code and data files. 2/ Automation of all computations, tables and plots, which practically requires the use of para-modeling tools such as script languages: Bash, Perl or the MS-DOS batch files. 3/ Literate programming, as the key to facilitate up to date documentation. 4/ Portability, using only high-level, platform and operating system-independent languages, such as standard Fortran, Perl or GAMS.

This paper highlighted the difference of status between the model name and the model code. The name is a trademark that can be protected without decreasing transparency, even when the code is published. Moreover, letting someone have a thorough look at a model does not necessarily imply that this person is allowed to use it, modify it or redistribute it.

A case study explored the license of four climate policy integrated assessment models. It found that presently, the peer-reviewing system achieves less transparency and control than the free software/open source development model in which programs are reproduced before final release by a large number of anonymous external peers.

In the study, the smaller model had an academic open license, while the larger model had a more competitive-oriented business, restricted license. Potential trans-

[1] See for example the Concurrent Version System at <http://cvshome.org>.

parency remains, since anybody can get the model simply by asking. However, I find no reason to be satisfied with potential transparency when models source code can be published at low cost with today's information technology.

The second part of the paper discussed publishing the source code and data. The GNU GPL was presented as a convenient way to enable the source code to be freely used, modified and redistributed. The most straightforward way to achieve transparency is to place all files under the GPL and archive them to a public, anonymously accessible repository.

In practice, it may not be possible or judicious to place all files under the same license. Files re-used from other projects may be copyrighted by their respective owner. Diffusion of preliminary results may be restricted so that they don't leak outside the peer-reviewing system before publication. Finally, as discussed in this paper, the payoffs of opening the source may not be the same for different parts of a model.

4.2 Practical implications and future directions

Many journals such as *Science* or *Nature* now offer electronic online supplements to their paper edition. There is clearly an opportunity here to improve transparency and control of integrated assessment models. Scientific journals review, disseminate and archive the text of articles. How to extend these functions to the model code and data files? The social and technical infrastructure of a specific publication structure can be outlined. Researchers' networks such as the EFIEA[2], EMF[3] or the CISHDGC[4] will have the leading role, in collaboration with commercial scientific publishers such as Baltzer and its ESIAM[5] series.

Accepting manuscripts submitted along with an archive of the code and data is a simple first step. The archive does not need to be online. But an online archive has the advantage that it enables authors to upload directly their files without the need for editor's attention. Many automated systems for sharing files online already exist.

Dissemination is more delicate than archival. For convenience, it could be the authors' own responsibility to upload the archived files into the journal's repository and to define to whom they allow access rights. Clearly, it needs to be possible to set different access rights for Author, Co-Authors, Editor, Referees, Members of the journal's Scientific Association, and the rest of the world. And some differentiation between data, code and output files may be important.

Establishing trust in the dissemination system allows a third step, that is model reviewing. This could be organized as book reviewing. Book reviews are contributed to a dedicated section of the journal, on a less formal basis than for articles. If authors of a submitted manuscript have reached the point where the model is portable, that is they claim that an anonymous referee could easily reproduce their results, then they would certainly be very happy to have this claim tested. In this situation, it is probably

[2]The European Forum on Integrated Environmental Assessment is at <http://www.efiea.org.>

[3]The Energy Modeling Forum is at <http://www.stanford.edu/group/EMF>

[4]The Center for Integrated Study of the Human Dimensions of Global Change URL is at <http://hdgc.epp.cmu.edu/>

[5]The Electronic Series on Integrated Assessment Modeling is at <http://www.baltzer.nl/esiam/esiam.asp>

better to publish the referee's notes on their experience with the model alongside the paper.

The fourth step would be to establish some common infrastructure[6] to facilitate the open-source development of scientific models.

All of this would insure a huge improvement of transparency and control in integrated assessment models.

Acknowledgements

I thank Jean-Charles Hourcade, Richard Tol, David Keith, Barbara Bugosh, Richard Richels, Pierre Matarasso, Philippe Quirion, Tarik Tadzait and Abigaïl Fallot for useful advice on this work. All errors and opinions remain mine.

This research has benefited from a visit at the Center for Integrated Study of the Human Dimensions of Global Change, Departement of Engineering and Public Policy at Carnegie Mellon University.

References

[1] Joseph Alcamo, editor. *IMAGE 2.0: Integrated Modeling of Global Climate Change, with papers by The IMAGE Project*. Kluwer Academic Publisher, 1994. Reprinted from Water, Air, and Soil Pollution 76(1-2).

[2] Franck H. Clarke. *Optimization and nonsmooth analysis*. Université de Montréal, Centre de Recherche Mathématiques (CRM), complete and unrevised reprinting of the original text first published in 1983 by john wiley and sons, as the initial volume in the canadian mathematical society series of monographs and advanced texts edition, 1989. 310 pages.

[3] Tom Kram. The energy technology systems analysis programme: History, the ETSAP Kyoto statement and post-Kyoto analysis. In *Experts Workshop on Climate Change and Economic Modelling: Background Analysis for the Kyoto Protocol*. OCDE, September 17-18 1998.

[4] Josh Lerner and Jean Tirole. The simple economics of open source. Technical report, Harvard Business School, February 25 2000.

[5] Alan S. Manne and Richard Richels. *Buying Greenhouse Insurance: The Economic Cost of CO_2 Emissions Limits*. MIT Press, 1992.

[6] Rick Moen. Fear of forking. Linuxcare Featured Article published online, November 17 1999.

[7] M. Granger Morgan and Hadi Dowlatabadi. Learning from integrated assessment of climate change. *Climatic Change*, 34:337–368, 1996.

[6] See for example Sourceforge <http://sourceforge.net>.

[8] William D. Nordhaus. *Managing the Global Commons*. MIT Press, 1994.

[9] Jerome R. Ravetz. Integrated environmental assessment forum: developing guidelines for good practice. Technical Report WP-97-1, ULYSSES project, Darmstadt University of Technology, Germany, EAWAG, Human Ecology Division, Ueberlandstr. 133, CH-8600 Duebendorf, Switzerland, 1997. http://www.zit.tu-darmstadt.de/ulysses/eWP97-1.pdf.

[10] Eric S. Raymond. La cathédrale et le bazar. *Le Micro Bulletin*, 75:81–112, 1998. Traduction de Sébastien Blondeel. Also published online at <http://www.tuxedo.org/ esr/writings/cathedral-bazaar/cathedral-bazaar.html>.

[11] Jan Rotmans. Methods for IA: The challenges and opportunities ahead. *Environmental Modeling and Assessment*, 3:155–179, 1998.

[12] Stephen H. Schneider. Integrated assessment modeling of global climate change: Transparent rational tool for policy making or opaque screen hiding value-laden assumptions? *Environmental Modeling and Assessment*, 2(4):229–249, 1997.

[13] Ferenc L. Toth, Thomas Bruckner, Hans-Martin Füssel, Marian Leimbach, and Gerhard Petschel-Held. The tolerable window approach to integrated assessments. In O. K. Cameron, K. Fukuwatari, and T. Morita, editors, *IPCC Asia-Pacific Workshop on Integrated Assessment Models*. NIES.CGER, 10-12 March 1997.

Annex: Articles 1-3 of GNU General Public License v2.

Preamble. The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software–to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps: (1) copyright the software, and (2) offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

Scope (article 0.) This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

Articles 1 to 3.

1. You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

   You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

2. You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

   (a) You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

   (b) You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

   (c) If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the

user how to view a copy of this License. (Exception: if the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

3. You may copy and distribute the Program (or a work based on it, under Section 2) in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

   (a) Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (b) Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

   (c) Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

   The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

   If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.