Towards a manifesto for open simulation

J. Gary Polhill
Macaulay Institute, Craigiebuckler, Aberdeen. AB14 0PR. United Kingdom.
g.polhill@macaulay.ac.uk

Bruce Edmonds
Centre for Policy Modelling, Manchester Metropolitan University, Aytoun Building, Aytoun Street. M1 3GH. United Kingdom.
bruce@edmonds.name

Abstract

The difficulties associated with adequately describing agent-based social simulation software in scientific journals necessitate the establishment of norms for making the software more accessible. We consider here norms for documentation and licensing that outline a manifesto for open simulation, which we hope the community will sign up to. Licensing in particular has a number of thorny issues associated with it: the default under the Berne Convention turns software into a black box, raising a question over whether it can form a legitimate part of the domain of scientific discourse, or for that matter, of open and accountable government. Researchers must therefore take action to ensure their software is not released under an inappropriate licence. We recommend that a manifesto be produced based on the proposals in this paper, and call upon researchers to (a) sign up to that manifesto; (b) implement its proposals; (c) comment on licences when reviewing articles and proposals; (d) encourage others to do the same.

Keywords: Agent-Based Social Simulation, Replication, Software Licences, Documentation, Archiving.

1. Introduction

Even the best social simulations are complex objects whose meaning and import are difficult to discern. One may read a description of a simulation and get a vague idea of its properties but, like mathematics, one only thoroughly understands a simulation when one has pulled it apart and played with it. Despite this fact, simulations have become part of the scientific discourse and are starting to enter the more general, public, discourse. In public domains simulations are often communicated without deep hands-on understanding, relying instead upon the skill and integrity of the community who created them. In this sense they are like mathematical models, simultaneously opaque to most and relied upon by many. However, unlike mathematical models, they facilitate attractive and accessible animations of their results—making them seem more transparent than they are and persuasive to a degree quite separate from their validity.

For these reasons it is important that simulations are openly accessible to others so that they can be checked, compared and improved upon. In this way they can be passed-on and developed by a series of independent researchers resulting in objects that are reliable and thoroughly understood (as far as this is completely possible). Indeed the simulations can be said to be *evolved* by the community of simulators. The easier it is to download, reimplement, compare, alter and try out the simulations, the more productive this process will be and the better its results. Some of these issues are discussed by more general calls to the social simulation community looking at open content issues (Schweik, Evans and Grove 2005) and understanding what simulations do (Alessa, Laituri and Barton 2006).

In a democratic society where simulations are used as an input to the formulation of policy there is a further reason why the simulations should be open. A simulation is not value-free but implicitly encodes its authors' assumptions about the phenomena it aims to represent. If a simulation forms part of a political process wherein it is used to help formulate policy, then it is essential that the simulation be open to inspection to reveal what assumptions and unexpected behaviours it might otherwise hide. Whilst we think it unlikely that the general public will often wish to inspect simulations in detail, the ability of researchers of differing views to probe simulations and publish the results means that the public can be better protected and the political debate around the issues connected to the simulation raised to a higher standard. Similar points are raised about medical software by Carnall (2000) in an editorial to the British Medical Journal.

Finally there is an issue of equality of access. De Laat (2005, pp. 1529-1530) notes that a 'private' licensing regime, protective of intellectual property, excludes outsiders, whilst a public regime is inclusive. Like other parts of the scientific literature simulations embody knowledge. To be sure this knowledge is often implicit, tentative and only partially understood (especially, dare we say it, in social simulation), but nonetheless the process they have gone through in terms of development, refinement, verification, replication and validation results in objects that are worth something. At the moment social simulations are worth relatively little but sometime in the future the situation might be different and they might afford the user some advantage in terms of understanding, and hence managing, aspects of society. Whilst there is an argument for the temporary private exploitation of discoveries, the majority of this work is publicly funded and should probably treated like most scientific knowledge—that is the greatest general benefit is gained by its free distribution and use. Further, Forero-Pineda (2006) notes (p. 816) that scientists from developing countries need access to frontier knowledge to facilitate modernisation and prosperity.

To summarise, there is a need for simulations in the public and academic domain to be:

- *Accessible* to active experimentation by independent third parties;
- *Transparent* as to the detail of their implementation and principles of operation;
- *Comparable* to each other in terms of their properties and resultant behaviour.

This manifesto considers some ways in which the above aims can be facilitated and calls upon social simulators to promote the practice of open simulation by implementing its suggestions and encouraging other to do likewise. In the context of increasing protection of intellectual property in the academic sector (Sampat, 2006), this is bound to be contentious. The key, however, is that the needs of Science as a domain of practice are not compromised, and we seek to address these issues through basing the discussion on these requirements.

2. The Adequate Documentation of Simulations

The first and most obvious way in which simulations can be made accessible is their adequate description when presented. Obviously such description is an open-ended task and can be achieved to different degrees. In this sections we discuss a number of different aspects of such description.

The most basic level of such a description is that which makes possible its reimplementation in all significant aspects. If the simulation is implemented using a commonly-used programming system then simply making the code available achieves this basic level. This is easy to do since even if the paper or presentation could not include all the detail of a simulation's code a pointer to the code on a suitable web-site or archive is easy to arrange. Of course, this does not make the understanding of the code or its reimplementation at all *easy* so that it is unlikely that most readers, even interested, expert simulators, would avail themselves of this. However, if there was a later dispute about the details this would help it be resolved.

Clearly more than this basic level is necessary if the reimplementation and improvement of code is to be at all *facilitated*. Reading and understanding raw code is extremely time-consuming and difficult. Explanation and illustration of key aspects of a simulation can greatly facilitate its accessibility. How this might be achieved best for each simulation is difficult to specify—there are a plethora of competing methods and standards for code documentation[1], and it is unlikely that any one will be best for all simulations. Grimm et al. (2006) do have a suggested standard with particular relevance to agent-based models. This is focused on a higher-level description than documentation of code. Nevertheless, there are some general principles that can be suggested, including:

- That the main processes and structures should be described in narrative form;
- That complex parts of simulations should be described in several different ways, for example as a natural language narrative *and* pseudo-code *and* illustrated using diagrams;
- That the following should be somehow covered: any permanent data-structures, the temporal structure indicating how events or agents are executed, the parameters of the simulation, key algorithms, and which code variations are used;
- That, regardless of the adequacy of the above, the unaltered code should be available to check details and as a matter of record.

However carefully one describes the design of a simulation, mistakes and complexities abound in all programming—it is simply impossible to anticipate the effects and interactions of all design decisions that one makes. This is vividly shown when one seeks to reimplement even the simplest of simulations in another programming language, library or framework (Edmonds and Hales 2003; Axtell, Axelrod, Epstein and Cohen 1996; Bigbee, Cioffi-Revilla and Luke 2005; Galan and Izquierdo 2005). Such efforts have typically involved considerable interaction between those conducting the reimplementation and the original authors to resolve ambiguities and implementation details. Thus in addition to information about the design of simulation, one also needs a lot of information about the behaviour of the simulation so that mistakes and unexpected implementation properties can be identified and maybe corrected. The following is helpful in this regard:

- A description in narrative form of the sort of behaviour that is observed when running the simulation;
- Graphs of simulation outcomes;
- Both summary results concerning behaviour that seems to be general to a wide range of parameter settings as well as detailed examples of specific runs and settings;

[1] Various organisations have devised relevant standards (judging from the titles), often available only at great expense; though Jose & Viswanathan pointed out in 1992 that documentation ‘standards’ are often guidelines rather than stipulations. Here are some examples, in chronological order:

NASA-STD-2100-91 “NASA Software Documentation Standard” [1991]
(http://satc.gsfc.nasa.gov/assure/docstd.html).

IEC 61506 “Documentation of Application Software” [1997]
(http://domino.iec.ch/webstore/webstore.nsf/artnum/021929)

IEEE 1016 “Recommended Practice for Software Design Descriptions” [1998]
(http://ieeexplore.ieee.org/xpl/tocresult.jsp?isNumber=16019)

BS ISO/IEC 6592:2000 “Information Technology. Guidelines for the documentation of computer-based application systems” [2000]
(http://www.standardsdirect.org/standards/standards3/StandardsCatalogue24_view_23963.html)

ISO/IEC 18019 “Guidelines for the Design and Preparation of User Documentation for Application Software” [2004]
(http://www.iso.org/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=30804).

- A set of numerical summary results that can be used to check the accuracy of any reimplementation (in which case how the statistics are obtained needs careful description);
- A summary description of the expected effects of changing various parameters, for example, which parameters are particularly sensitive;
- Ideally the simulation should be available to be rerun in a manner that makes it easy to obtain more results using parameters chosen by the reader;
- Often it is necessary to be able to change the original code to make fresh simulation experiments in order to track down differences in behaviour, for example by changing the selection method in an evolutionary simulation or to eliminate a source of chance.

This does not quite complete the information that is necessary to completely understand a simulation however, for the embedding of the simulation within the scientific and cultural processes is also important. Thus it is helpful to also include:

- Which other simulations it relates to, and in particular references to previously published simulations that are antecedents of the one described;
- A description the mental model of the simulator—in other words, which elements of the simulation are *representative* of the core intention of the modeller and which are implementation details that were added merely to get it to run;
- The justification (if any) for the design choices made in the simulation, so that the original purpose of parts of a simulation can be understood;
- Which aspects of the observed simulation behaviour are considered by the modellers to be significant and which not (e.g. those attributed to "noise");
- A description or reference to any other theories and models that the simulation is supposed to encapsulate or illustrate.

In summary, adequately describing a complex and dynamic object such as a social simulation is not easy—it requires multiple descriptions using many different methods and covering many different aspects. Although formal lists or systems of simulation description may help ensure that a minimum level is achieved in publications, it is likely that the best description will need to include additional methods that are particularly suited to particular simulations. As long as the description is suitably structured (e.g. via the use of appendices for technical detail) then more is generally better. At the same time, documentation is a time-consuming activity that many researchers cannot afford to undertake. There are two approaches that can be adopted to address this. One is to reduce the need for manually documenting software by finding ways to construct documentation automatically from code. Another is to include in grant proposals a provision for documentation of any involved software, an activity that could be conducted by a contracted professional.

3. Facilitating Access

The most trivial aspect of making a simulation open is ensuring access to the relevant documents concerning it. To be useful this needs to be more than just linking the code from a web page, because the code is just part of a simulation's documentation. Ideally, the following should be accessible:

- The code itself (preferably annotated);
- Instructions on how to use it, including software needed for compilation, necessary libraries and where to get them from, and simple instructions about running it;
- Documents which describe the simulation design and specification;

- Documents which describe in qualitative terms the behaviour and results from running the simulation;
- Sample data extracted from runs of the code with given parameters so others can check they have re-implemented a simulation correctly;
- Papers which are relevant to the code, including sources for the design ideas, papers where the code is used, papers which describe the context of the simulation effort, and papers which extend the code;
- Any other simulations which related to the code, including replications and extensions;
- The individuals and institutions that have been involved in developing it.

This basic act of making information about simulations freely accessible can be seen as an extension of the call to "free the academic literature" made by Stevan Harnad and others (Harnad 1998)[2]. Of course if the above information is scattered all over the internet and many journals, then, *firstly*, not many people will find it in the first place and, *secondly*, if they do they will have great difficulty in finding the other documents listed above. Thus an openly queryable database that provided the links between these different documents would greatly facilitate the accessibility and openness of such simulations. At the moment it is a journal paper or institutional website that seems to play this role, but this does not provide the links between related institutions and related documents in very structured ways. Thus perhaps some sort of structured "open-archive"[3] would be helpful, maybe on the lines of the publicly funded data archives (e.g. the UK Data Archive[4]).

4. The Appropriate Licensing of Simulation Software

One of the most important aspects of any piece of software, besides what it actually does, is the licence. The licence determines the relationship the end-user community will have with the software, specifying who can use it, and what they can do with it. For commercial software, the purpose of the licence is to ensure that the company developing the software are able to maximise their profits from it: users are typically prevented from copying, adapting or distributing the software, and copying or distributing any adaptation.[5] Such protection is the default assumed under law in those countries adopting the Berne Convention, in which a created work is considered protected by copyright as soon as it exists.[6] Computer programs are usually considered as 'literary works' for legal purposes, though some of the rights enjoyed by authors are not necessarily extended to creators of software.

Whatever the default legal assumptions, software is not only written with a view to making money. In science, software simulations are increasingly forming part of the explanation process, and related to this, output from software simulations (especially agent-based social simulations) may be used as a basis for advice to policy-makers. There is also the moral question of whether citizens should be prevented from using and inspecting software developed using funds from their taxes. However, unless the copyright owner specifically issues a licence allowing people to copy, adapt and distribute their software, the default assumptions mean that no-one can do so without committing an infringement of copyright. In many cases, the owner of the copyright will be, by default, the employer of the creator of the software rather than necessarily the creator themselves.

[2] For an extensive introduction to this issue see: http://www.ecs.soton.ac.uk/~harnad/#Openaccess

[3] As in the "Open Archives Initiative", http://www.openarchives.org/

[4] http://www.data-archive.ac.uk/

[5] End-user licence agreements also often prohibit disassembling or decompiling the software to see how it works, though (at least in European law), there are circumstances in which this is allowed (Directive 91/250/EEC) even if the licence agreement stipulates against it.

[6] See http://www.wipo.int/copyright/en/faq/faqs.htm

As a result, the permission of the employer may be needed to release the software under an appropriate licence.

This section discusses the rights that should be made available in software licences for these kinds of application. Of course, in many scientific proposals, consideration is often given to the scope for commercial exploitation of the research, with many funding bodies looking favourably on those proposals with good such prospects. In attempting to establish acceptable norms for software licences in agent-based social simulation, we may not wish to adversely affect our chances of winning funds. Gambardella and Hall (2006) consider some of the issues of open-source licensing with respect to commercial exploitation. Nonetheless, for software used in scientific research there are arguably some basic rights that need to appear in the licence, without which the work associated with the software should not be considered a legitimate part of the scientific domain. There are also arguments for suggesting similar rights for software used as a basis for advice to policy-makers, and for software developed using public funds.

Though we draw on much of the material from the Free Software Foundation, it is not the place of this document to stipulate any particular licence as being the only one that is acceptable. Instead, the purpose is to present criteria on the basis of which a licence agreement may be accepted or rejected for the purposes of including a piece of software in a certain domain. For example, in the case of scientific software, there is an argument for suggesting that the licence should be given as part of the review process for any academic paper associated with the software, since if the paper relies on software with an unacceptable licence this is a potential basis for rejection.

Some terminology in the area of licensing needs careful unpicking, as there is a great deal of confusion.[7] The following uses the work of Richard M. Stallman and the Free Software Foundation as a basis. The most difficult word is the word 'free', which can mean both "zero economic cost" and "giving freedoms" (Stallman 2002b). It is the latter with which we are most concerned here: the freedoms that are required for the various uses we are considering—a zero cost piece of software could have a commercial-style licence preventing copying and modification. The term 'free software' has a very specific definition (Stallman 2002a) pertaining to particular freedoms/rights such as those that appear in the GNU General Public Licence. It is quite possible, within the terms of that licence to pay money for 'free software', though the purchaser is then entitled to copy the software to whoever they want. The confusion over the term 'free' led in part to the term 'open source'[8]. A strict interpretation of 'open source' is that it pertains to a specific right—that of inspecting the source code. Though this is an important right as argued later, it is not only right that is required here. An open source licence could be unacceptable in certain circumstances if it prevented other freedoms that are necessary. However, the Open Source Initiative has a certification mark for approved licences based on its 'Open Source Definition', which has conditions that are more wide-ranging than this strict interpretation of the term[9]. A third term worth understanding is 'copyleft'. Copyleft is a condition inserted into a software licence that prevents redistributors of the software and any modifications of it from adding any additional restrictions in their licences. Whilst this could be seen as a restriction on the freedom of the end-user, it can also be seen at the community level as perpetuating the freedoms given by the original author. It would be rather irritating having released some code under a non-copyleft licence, to later find that another party had released a modification with a useful enhancement or bug fix under a proprietary licence that prevented anyone from seeing what had been done. Lin, Ko, Chuang and Lin (2006) note that documentation of modifications provides

[7] More information on the meaning of various terms pertaining to software licences, including those discussed here, can be found at http://www.gnu.org/philosophy/categories.html

[8] The Open Source Initiative website explains more about this term, and provides its own arguments against the term 'free'. See http://opensource.org/advocacy/free-notfree.php.

[9] See http://opensource.org/docs/definition.php.

useful feedback to the original developers. From a scientific point of view, while journals do not stipulate licensing criteria for software involved in publications, copyleft protects the author from criticisms of closed modifications to their work.

Proprietary software licences turn the software into a black box, which the user must trust to function as advertised. There is a great deal of evidence that such trust is entirely misplaced even in the commercial sector, as maintenance and update contracts, and frequent urgent requirements to install revisions for security reasons will testify. However, the issue of trust is of paramount importance in the world of science, as scientists are not in the business of trusting that each others' theories are correct: repeatability and verifiability are pillars of the scientific epistemological framework. Trust is also an issue in a free, open, democratic society—governments do not have the right to assume the trust of their citizenship, and should expect their decisions to be open to scrutiny. There is therefore a question over whether 'black box' software could legitimately form part of an open and accountable decision-making process. Software models are already used extensively in Government, not least in the area of climate change. If citizens are to be asked to make great sacrifices for the sake of the environment, should they not have the right to check the evidence fully and decide for themselves?

Clearly, therefore, there are certain cases where black box licences are inappropriate. It may be argued that in opening such black boxes, there is the potential for unqualified individuals to use the software inappropriately, or draw incorrect conclusions from their inspections of it. However, it is unlikely that most people will find the time to inspect what may be tens of thousands of lines of code, and thus the task will in any case be left to individuals who are qualified to undertake the task on their behalf, such as other researchers, investigators or journalists. Such specialists will then disseminate their discoveries to their target audiences using appropriate language. The important thing is that they are not prevented from doing so by overly-restrictive licences, lest software programs become the tea-leaves and tarot cards of a new generation of (cyber-)prophets.

The following rights are suggested as stipulations for a piece of software that forms the basis of scientific research or advice to policy-makers:

- *The unrestricted right to run the software*. Though this may seem trivial, the right to run the software enables both scientists and investigators to check that it works as specified. However, the *way* in which the software is run can be subject to copyright regulation. In the UK, section 29 paragraph 4A of the copyright legislation specifically states that except when using the program as "entitled" (section 50BA paragraph 1), it is considered unfair "to observe, study or test the functioning of a computer program in order to determine the ideas and principles which underlie any element of the program". In the case of scientific software, this is too prohibitive a restriction, since in the absence of any other rights, the ideas and principles underlying any element of the program are critical to its scientific credentials, and should thus be open to scrutiny. Similar arguments apply to software used as part of the policy-making process.
- *The right to inspect the source code*. The argument for this is on the basis that in mathematical modelling the formulae and derivations are always shown in a scientific paper, allowing them to be checked by other readers. This is not practical in the case of source code, where several thousand lines are involved. It is not necessarily the case that the source code should be available for free, but imposing a financial cost on acquiring the source code presents an impediment to checking the work of others that perhaps should not be there. For software involved in the policy-making process, the right to inspect the source code is an important part of an open approach to scrutiny of the decision-making process, not least to voting itself.[10]

[10] The Australian Capital Territory Electoral Commission, for example, released the source code for its electronic voting system (http://www.elections.act.gov.au/Elecvote.html). However, the Electronic Frontier

- *The right to re-implement the model*. Not necessarily preventable by a software licence, but potentially an issue for software patents, the right to re-implement a model is of paramount importance in the scientific process, as testified by various authors whose re-implementations of original models challenge the conclusions derived from them (e.g. Edmonds and Hales 2003; Galan and Izquierdo 2005).
- *The right to modify the source code*. This is to enable checking for what might be termed the 'algorithmic sensitivity' of the model. For example, simply changing the order in which a supposedly arbitrarily-ordered list is processed has been shown to change the output of a model (Polhill, Izquierdo and Gotts 2005—though in this case, not significantly).
- *The right to distribute the modified version*. If those modifications are then used in a scientific publication, the same norms would apply as to the original software: one should not be allowed to criticise someone else's work without also allowing that criticism to be open to scrutiny. This requires rather more complex licensing arrangements, as the modified source code would need to be made available without any infringement of the original developers' intellectual property rights. These issues are addressed by licences such as the GNU General Public and Academic Free Licences. However, the *right* to distribute the modified version is not a stipulation that it *must* be distributed under certain conditions (e.g. publication of a paper based on the modified version). For software involved in a scientific publication or a policy-making process, such a stipulation should perhaps be made. None of the licences reviewed in Appendix B below make any such stipulation.
- *Copyleft protection*. Software licences for scientific and policy-making purposes should protect the original authors by stipulating that modifications must be issued under a licence that gives all the above rights to end-users.

Since licence agreements can be subject to change and national boundaries, these rights should be explicitly stated as being *irrevocable, world-wide, nonexclusive,* and (preferably) *royalty-free.*

5. Conclusion

Clearly the licensing and documentation of software can facilitate or prohibit good scientific practice. Obfusticated, uncommented code with no design documentation or user guide is the most extreme form of undocumented software, and would be an utterly useless contribution to knowledge. With licensing in particular, legislation aimed at protecting businesses' intellectual property is overly protective when it comes to software that is to be considered part of science. We have considered above the rights that we believe must necessarily appear explicitly in scientific software licences for the software to be considered a legitimate part of the scientific domain of discourse. Such arguments also apply to software used as part of open and accountable governance. In an appendix to this paper, we consider a set of popularly-used licences against these criteria. In our opinion, the rights we have outlined describe what should be standard practice for those in the social simulation community, and the burden of proof should be on those who do not come up to this standard to justify why their simulation should be exempt. We hope that this practice will become the norm for social simulators—that simulators will feel that such practice is simply part of the job of being a researcher who uses simulation as a tool.

We recommend that JASSS prepare a manifesto for open simulation based on the contents of this paper. We would then call upon researchers to:

- add their name to those who agree with that manifesto;

Foundation reports on its website that some votes in the November 2004 US elections were cast on electronic voting machines with black box licences (http://www.eff.org/Activism/E-voting/).

- implement its proposals;
- comment on software licensing when reviewing papers and proposals;
- encourage others to do so.

6. References

ALESSA L N, Laituri M and Barton M (2005). An “all hands” call to the social science community: Establishing a community framework for complexity modeling using agent based models and cyberinfrastructure. *Journal of Artificial Societies and Social Simulation* 9 (4). http://jasss.soc.surrey.ac.uk/9/4/6.html.

AXTELL R, Axelrod R, Epstein J and Cohen M D (1995). Aligning simulation models: A case study and results. *Computational and Mathematical Organization Theory* **1** (1), pp. 123-141.

BIGBEE A, Ciofii-Revilla C and Luke S (2005). Replication of Sugarscape using MASON. In Troitzsch, K. G. (ed.) *Representing Social Reality: Pre-proceedings of the Third Conference of the European Social Simulation Association, Koblenz, September 5-9, 2005*. Koblenz: Verlag Dietmar Fölbach. pp. 6-15.

CARNALL D (2000). Medical software’s free future. *British Medical Journal* **321**, p. 976.

DE LAAT P B (2005). Copyright or copyleft? An analysis of property regimes for software development. *Research Policy* **34**, pp. 1511-1532.

EDMONDS B and Hales D (2003). Replication, replication and replication: Some hard lessons from model alignment. *Journal of Artificial Societies and Social Simulation* **6** (4). http://jasss.soc.surrey.ac.uk/6/5/11.html.

FORERO-PINEDA C (2006). The impact of stronger intellectual property rights on science and technology in developing countries. *Research Policy* **35**, pp. 808-824.

GAMBARDELLA A and Hall B H (2006). Proprietary versus public domain licensing of software and research products. *Research Policy* **35**, pp. 875-892.

GALAN J M and Izquierdo L R (2005). Appearances can be deceiving: Lessons learned re-implementing Axelrod’s ‘Evolutionary Approach to Norms’. *Journal of Artificial Societies and Social Simulation* 8 (3). http://jasss.soc.surrey.ac.uk/8/3/2.html.

GRIMM V, Berger U, Bastiansen F, Eliassen S, Ginot V, Giske J, Goss-Custard J, Grand T, Heinz S K, Huse G, Huth A, Jepsen J U, Jørgensen C, Mooij W M, Müller B, Pe’er G, Piou C, Railsback S F, Robbins A M, Robbins M M, Rossmanith E, Rüger N, Strand E, Souissi S, Stillman R A, Vabø R, Visser U and DeAngelis D L (2006). A standard protocol for describing individual-based and agent-based models. *Ecological Modelling* **198** (1-2), pp. 115-126.

HARNAD S (1998). On-Line Journals and Financial Fire-Walls. *Nature* **395** (6698): 127-128. http://cogprints.soton.ac.uk/documents/disk0/00/00/16/99/index.html

JOSE J M and Viswanathan T (1992). Software documentation and standards. *Annals of Library Science and Documentation* **39** (4), pp. 123-133.

LIN Y H, Ko T M, Chuang T R and Lin K J (2006). Open source licenses and the creative commons framework: License selection and comparison. *Journal of Information Science and Engineering* **22** (1), pp. 1-17.

POLHILL J G, Izquierdo L R and Gotts N M (2005). The ghost in the model (and other effects of floating-point arithmetic. *Journal of Artificial Societies and Social Simulation* **8** (1). http://jasss.soc.surrey.ac.uk/8/1/5.html.

SAMPAT B N (2006). Patenting and US academic research in the 20th century: The world before and after Bayh-Dole. *Research Policy* **35**, 772-789.

SCHWEIK C, Evans T and Grove J M (2005). Open source and open content: A framework for collaboration in social-ecological research. *Ecology and Society* 10 (1): 33. http://www.ecologyandsociety.org/vol10/iss1/art33/

STALLMAN R M (2002a). Free software definition. In Gay J (Ed.) *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Boston, USA: GNU Press, pp. 41-43.

STALLMAN R M (2002b). Why “free software” is better than “open source”. In Gay J (Ed.) *Free Software, Free Society: Selected Essays of Richard M. Stallman*. Boston, USA: GNU Press, pp. 55-60.

7. Legislation

UK

Copyright Rights in Performances. Publication Right. Database Right. Unofficial Consolidated Text of UK Legislation to 31st December 2003. HMSO.

http://www.patent.gov.uk/cdpact1988.pdf

Europe

Council Directive 91/250/EEC of 14 May 1991 on the legal protection of computer programs. Official Journal L 122, 17/05/1991 P. 0042-0046.

http://europa.eu.int/eur-lex/lex/LexUriServ/LexUriServ.do?uri=CELEX:31991L0250:EN:HTML

International

WIPO Copyright Treaty

http://www.wipo.int/treaties/en/ip/wct/trtdocs_wo033.html

http://www.wipo.int/treaties/en/ip/wct/pdf/trtdocs_wo033.pdf

Berne Convention for the Protection of Literary and Artistic Works

http://www.wipo.int/treaties/en/ip/berne/trtdocs_wo001.html

http://www.wipo.int/treaties/en/ip/berne/pdf/trtdocs_wo001.pdf

UNESCO Universal Copyright Convention

http://www.unesco.org/culture/laws/copyright/html_eng/page1.shtml

http://www.unesco.org/culture/laws/copyright/images/copyrightconvention.rtf

8. Acknowledgements

This work was funded by the Scottish Executive Environment and Rural Affairs Department.

9. Appendix A: a quick HOWTO guide to licensing a simulation

An explanation of how to licence software using the GNU General Public Licence is found at this website: http://www.gnu.org/licenses/gpl-howto.html. Certain general principles can be abstracted from this:

- Each file in the source code should contain the following, at the top of the file, or as close as possible to it (in comments):
    - A statement of copyright "Copyright (C) <year(s)> <name of copyright owner(s)>". Note that the copyright owner could legally be your employer if you have signed a contract to that effect. This means you will need permission from your employer to release the code under the licence. A year should be included in the notice for each year in which the software was released.
    - A brief statement of what the licence allows you to do. If you have used code from other programs, this should include any restrictions from the licences of those programs. You may also need to preserve their copyright notices as a condition of the licences, so make sure you understand the terms of the licences carefully.
    - A disclaimer of warranty. The GNU GPL how-to page recommends the following text: "<program name> is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE." A reference is then made to the main LICENCE file in the directory for details.
    - A statement of where to find the full licence agreement should a copy of it not appear in the source code directory.
- The directory containing the source code should contain the following files:
    - A LICENCE file containing the text of the licence agreement.
    - A ChangeLog file (or similar) to record the changes made to the code. The format for the ChangeLog file can be found at the following GNU web page: http://www.gnu.org/prep/standards/html_node/Change-Logs.html#Change-Logs. Keeping some sort of record of what has been done is important in ensuring that authors' reputations are protected from modifications to their work done by others.
- If necessary, somewhere, either in the directory containing the source code, or on the webpage on which it is released, it should be possible to obtain a statement from your employer that you have permission to release the software under the licence you have given it.

10. Appendix B: a review of various licences

Here we consider a few of the more popular licences under which software is released against the criteria set out in the paper. None are an exact fit. Each licence is considered in a table, which for each of the criteria gives a summary statement of whether it is met ('Yes' if it is, 'No' if it isn't, and '?' if there is some doubt) and quotes relevant sections of the licence. The criteria are: 'Run?'—does the licence explicitly state that there are no constraints to how the user may run the program? 'Inspect?'—does the licence stipulate that the source code be provided? 'Reimplement?'—does the licence explicitly allow you to use any patents in the software? 'Modify?'—does the licence explicitly permit you to create derivative works from the software? (Lin et al.'s (2006) comparison of licences also notes whether documentation of modifications made is stipulated.) 'Distribute?'—does the licence explicitly permit you to distribute the derivative works? 'Copyleft?'—does the licence force you to give at least the same freedoms in released derivative works? Note that this review has not been conducted by qualified legal personnel. A more comprehensive review of licences (with a difference of emphasis) can be found at http://www.gnu.org/licenses/license-list.html, and some of the licences reviewed here are also reviewed by Lin et al. (2006) against an intersecting set of criteria.

Licence	Open Source Definition compliant licence	
Link	http://opensource.org/docs/definition.php	
Example Software	A list of compliant licences is given at http://opensource.org/licenses/.	
Run?	?	"The license must not restrict anyone from making use of the program in a specific field of endeavor." If science is a field of endeavour, and it is generally accepted that using the program in that field entails the unconstrained right to run the program, then an OSD compliant licence would have a 'Yes' entry here. However, the clause is mainly aimed at ensuring commercial exploitation is not prohibited by the licence.
Inspect?	Yes	"The program must include source code …" with the following qualification: "Where some form of a product is not distributed with source code, there must be a well-publicized means of obtaining the source code for no more than a reasonable reproduction cost."
Reimplement?	No	Patents are not mentioned in the OSD.
Modify?	Yes	"The license must allow modifications and derived works…"
Distribute?	Yes	"…and must allow them to be distributed under the same terms as the license of the original software." Again, there is a small qualification: "The license may restrict source-code from being distributed in modified form *only* if the license allows the distribution of "patch files" with the source code for the purpose of modifying the program at build time."
Copyleft?	No	There is no stipulation that modified or derived works *must* be licensed under the same terms. The stipulation is only on the licensor to *permit* derived works to be distributed under the same terms.
Notes	The Open Source Definition is a series of constraints that must be adhered to by compliant licences, rather than a licence itself.	

Licence	GNU General Public Licence	
Link	http://www.gnu.org/licenses/gpl.html	
Example Software	Swarm 2.2	
Run?	?	"Activities other than copying, distribution and modification are not covered by this License; they are outside its scope." This suggests that the unrestricted right to run the program is not explicitly part of the licence, however, the licence then states: "The act of running the Program is not restricted…" On balance, this is probably a 'No'.
Inspect?	Yes	You must accompany the distribution of object code with "complete corresponding machine-readable source code," or provide a written offer to do so "for a charge no more than your cost of physically performing source distribution" or access to the written offer you have received.
Reimplement?	?	The licence does provide in the event of a patent infringement that "if … conditions are imposed on you (whether by court

		order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License.” However, the licence does not explicitly license you to use any patents owned by the licensor. It is not clear if a patent infringement of a distinct piece of software reimplementing the licensed software would cause the GPL to become invalid for the latter.
Modify?	Yes	“You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program.”
Distribute?	Yes	“You may copy and distribute verbatim copies of the Program's source code as you receive it … and copy and distribute … modifications or work [based on the Program].”
Copyleft?	Yes	“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.”
Notes	The “machine-readable” wording of the licence does not explicitly exclude deliberately obfusticated source code (e.g. automatically replacing meaningful identifier names with meaningless ones), which is a stipulation of OSD compliant licences.	

Licence	GNU Lesser (or Library) General Public Licence	
Link	http://www.gnu.org/licenses/lgpl.html	
Example Software	Swarm 2.1.1	
Run?	?	As per GNU GPL, though note that software using the library must be released under a licence that permits “reverse engineering for debugging,” which may tip the balance back to ‘Yes’.
Inspect?	Yes	As per GNU GPL, and see notes to GNU GPL.
Reimplement?	?	As per GNU GPL
Modify?	Yes	As per GNU GPL
Distribute?	Yes	As per GNU GPL
Copyleft?	?	It depends on whether the software is a derivative work, or a work that uses the library. The latter is “A program that contains no derivative of any portion of the Library, but is designed to work with the Library by being compiled or linked with it,” about which the licence says that “you may also combine or link a "work that uses the Library" with the Library to produce a work containing portions of the Library, and distribute that work under terms of your choice.” For derivative works and modifications to the library, “You must cause the whole of the work to be licensed at no charge to all third parties under the terms of this License.”
Notes	Provision is made within the LGPL to convert copies of the library and derivative works thereof to the GPL.	

Licence	Revised and Original BSD Licences	
Link	Revised: http://www.xfree86.org/3.3.6/COPYRIGHT2.html#5 Original: http://www.xfree86.org/3.3.6/COPYRIGHT2.html#6	
Example Software	Repast	
Run?	No	The unrestricted right to run the software is not stipulated. You are given the right to use the software, but the definition of ‘use’ could be interpreted as meaning only fair use.
Inspect?	No	There is no stipulation that the source code be provided.
Reimplement?	No	There is no statement made about patent licences.
Modify?	Yes	Modification is permitted.
Distribute?	Yes	Distribution of modifications is permitted.
Copyleft?	No	There is no stipulation as to the licence for derivative works.
Notes		

Licence	Mozilla Public Licence	
Link	http://www.mozilla.org/MPL/MPL-1.1.html	
Example Software	Protégé	
Run?	No	The right to use the software is not explicitly unrestricted.
Inspect?	Yes	“The Source Code version of Covered Code may be distributed only under the terms of this License,” and “any Modification which You create or to which You contribute must be made available in Source Code form under the terms of this License.”
Reimplement?	No	“No patent license is granted: … separate from the Original Code”
Modify?	Yes	The right to modify is explicitly stated.
Distribute?	Yes	The right to distribute is explicitly stated.
Copyleft?	No	“You may create a Larger Work by combining Covered Code with other code not governed by the terms of this License,” where a “Larger Work” is defined as “a work which combines Covered Code or portions thereof with code not governed by the terms of this License.”
Notes		

Licence	Academic Free Licence	
Link	http://www.opensource.org/licenses/afl-3.0.php	
Example Software	MASON	
Run?	No	The licence states that “You may use the Original Work in all ways not otherwise restricted or conditioned by this License *or* by law,” [our emphasis] which may unintentionally fall foul of

		the issue raised with UK law above.
Inspect?	Yes	"Licensor agrees to provide a machine-readable copy of the Source Code of the Original Work along with each copy of the Original Work that Licensor distributes." As per the OSD, this condition can be satisfied if the Licensor places the source code at a published conveniently accessible location.
Reimplement?	Yes	You are granted a licence "under patent claims owned or controlled by the Licensor that are embodied in the Original Work as furnished by the Licensor, to make, use, sell, offer for sale, have made, and import the Original Work and derivative works thereof."
Modify?	Yes	"Licensor hereby grants … You a … non-exclusive license … to modify … the Original Work."
Distribute?	Yes	"Licensor hereby grants … You a … non-exclusive license … to distribute … the Original Work and Derivative Works"
Copyleft?	Yes	The licence under which you distribute copies and derivative works must be one "that does not contradict the terms and conditions … in this Academic Free License."
Notes	There are several versions of the Academic Free Licence, with subtle differences between them. The above applies to version 3.	

Licence	Public Domain	
Link	http://en.wikipedia.org/wiki/Public_domain	
Example Software		
Run?	Yes	
Inspect?	Yes	
Reimplement?	Yes	Public domain means that no-one has any proprietary interests on the article in question.
Modify?	Yes	
Distribute?	Yes	
Copyleft?	No	
Notes	Public domain software is software that is not copyrighted. This is *not* the default, even if you have not explicitly written a copyright message in the material. The default, for signatories to the Berne convention, is that the material is copyrighted at the instant of creation.	