Open-Source Policy Modelling

Max Henrion

15 June 2006

Summary

Computer models are widely used by government to analyze, predict, and evaluate the benefits, risks, and costs of existing and proposed public policy and regulations. The US Office of Management and Budget (OMB) has issed a series of guidelines for the conduct of such analyses. Among other things, they require that all US regulatory agencies ensure that their analyses are transparent and reproducible.

This paper proposes extending these guidelines one step further:

> **All computer models used by government to evaluate or justify public policy should be *open source*: That is, the source program code in which the models are written should be publicly available for anyone to download, review, run, and modify.**

This proposal is inspired by the remarkable success of open-source software — such as Linux, Apache, My SQL, and Firefox — and other open collaborative ventures, such as Wikipedia, the online encyclopedia. Open-source projects may have some interesting lessons for policy modeling: There are important similarities and differences between existing open-source software and model for policy and risk analysis. Possible benefits of open-source policy modeling include improved transparency, fewer errors, and greater reusability and extensibility. It could also lead to an expanded, more collaborative development process with a wider range of contributors, building more constructively on each others' contributions. A key issue will be the fate of current proprietary models. There will be benefits and challenges for each kind of organization affected: government agencies, stakeholder groups (including industry associations, environmental and social justice groups, and community groups), universities and nonprofit think-tanks, and consulting analysts and firms.

To get a copy of the full 9-page paper, please contact Max Henrion
Henrion@Lumina.com
408-353-4891

Background

OMB’s Guidelines for good policy and risk analysis

In recent years, the Office of Management and Budget (OMB), has issued a series of guidelines for improving the quality of information and analysis used by government as a basis of public policy. OMB’s Data Quality Guidelines (OMB, 2001) provide standards of transparency and reproducibility for information and analysis used as a basis of regulations. Circular C4 (OMB, 2003) specifies methods for good regulatory analysis, including quantification of costs and benefits, probabilistic treatment of uncertainties, and sensitivity analysis, among others.[1] OMB (2004) extended the data quality guidelines with suggestions for the conduct of peer review. Most recently, the Proposed Risk Assessment Bulletin (OMB, 2006) provides technical guidance for risk assessments by the federal government. Most of these documents were developed under the leadership of John Graham, the Administrator of OMB’s Office of Information and Regulatory Affairs (OIRA) from 2001 until early 2006.

The Proposed *Risk Assessment Bulletin* (OMB, 2006) provides detailed guidelines for transparency and reproducibility:

> “A risk assessment report should … have a high degree of transparency with respect to data, assumptions, and methods that have been considered. Transparency will increase the credibility of the risk assessment, and will allow interested individuals, internal and external to the agency, to understand better the technical basis of the assessment.” (OMB, 2006)
>
> “Influential[2] risk assessments should be capable of being substantially reproduced. … As described in the OMB Information Quality Guidelines [OMB, 2001], this means that independent reanalysis of the original or supporting data using the same methods would generate similar analytical results, subject to an acceptable degree of precision. Public access to original data is necessary to satisfy this standard…” (OMB, 2006)

Computer models for policy analysis

The complexity of most regulatory policy analysis and risk assessments means that they almost invariably use computer models. Examples include cost-benefit analysis of traffic safety measures, risk analyses of proposed new drugs or food additives, assessments of suspected environmental carcinogens, or assessment of the effect of atmospheric emissions standards or emissions permit trading programs on air quality and its effects on human health.

Simple analyses are often implemented as spreadsheets, sometimes with an add-in package for Monte Carlo simulation, such as @Risk or Crystal Ball, for the probabilistic treatment of risk and uncertainty. Other analyses use visual modeling tools designed for policy analysis, such as Analytica, Extend, or iThink. The most complex models often use custom software, which is sometimes proprietary to the companies that developed them.

---

[1] I cannot but welcome most of these guidelines, having long advocated that policy analysts should borrow more extensively from the standard practices of science (Henrion, 1984). Granger Morgan and I had the temerity to propose “ten commandments for good policy analysis” (Morgan & Henrion, 1990) with which the OMB guidelines have a gratifying degree of overlap.

[2] OMB defines an assessment as *influential*, and so subject to this requirement, if it has potential impact of more than $500 million in any year, or “is novel, controversial, precedent-setting, or has significant interagency interest.” (OMB, 2001)

What is open-source software?

Source code means the program or model in the original language in which the programmer or modeller wrote it — such as C++ or spreadsheet formulas. *Open-source software* means software whose license lets anyone download, review, modify, improve, and redistribute its source code — in contrast to conventional proprietary software whose writers and publishers often to go great pains to make sure that no-one else can access the source code.

Open-source software products have been remarkably successful: Most of the software managing the Internet's infrastructure is open source. The Linux operating system now runs on perhaps 30% of computer servers; Apache has approaching 70% share of the web server market; about 44% of database systems use My SQL; a more recent entrant, the Firefox web browser is already up to about 14% penetration in the USA, and more elsewhere.

Open-source software is not necessarily *free*— that is, offered at no cost. Several companies, such as Red Hat Software and My SQL, have a successful business selling open-source software along with documentation, support, and certifications that provide additional value that people are willing to pay for. But, the ability for others to redistribute open source software limits the price they can charge for the software without the extra services.

The proposal

The kernel of this paper is the following proposal:

> **All computer models used by government to evaluate or justify public policy should be *open source*: That is, the source program code in which the models are written should be publicly available for anyone to download, review, run, and modify.**

This principle is applicable to policy models used by any public agencies at any level of government — local, state, regional, national, and international. It is relevant to all forms of public policy, including legislation, regulations, and budgets. However, the focus here is on OMB guidelines for federal regulatory agencies of the United States, and specifically OMB's Proposed Risk Assessment Bulletin (proposal issued in January, 2006).

Does this preclude proprietary tools, such as spreadsheets?

No. It is neither practical nor necessary to require that modelers abandon proprietary applications, such as Microsoft Excel, @Risk, Analytica[3], or, in the case of custom software, computer languages with proprietary compilers and interactive development environments. The important thing is that the source code of the model — for example, spreadsheet formulas, model equations, or procedural code — is accessible to read, run, and edit, and that the proprietary software is easily available and not prohibitively expensive.

What do OMB Guidelines say about this?

OMB's Guidelines, including the Data Quality Guidelines (OMB, 2001), Circular C4 (OMB, 2003), and the Proposed Risk Assessment Bulletin (OMB, 2006), call for assessments to be transparent and reproducible. They do not seem to address explicitly the issue of model source code. But, as a practical matter, it is hard to see how an analysis using a computer model could be transparent and reproducible without releasing its source code. In principle, model authors could provide specifications for the model in a natural language sufficiently detailed that the model could be rewritten. But, that would be a lot of extra work for the model authors and even more work for reviewers who would need to rewrite the model. In any case, without the

[3] As the originator and publisher of the proprietary modeling software, Analytica (Lumina, 2005), I admit a special interest here.

source code, it is impossible for reviewers to determine if any discrepancies in results are due to inaccurate specifications, errors in the original implementation, or errors in the reproduction. However, OMB recognizes that some models contain confidential information or may be proprietary, and admits those as compelling reasons not to publish models (to be discussed below).

Benefits and challenges

Transparency

In their replies to comments on the Data Quality Guidelines (OMB, 2001), OMB says:

> “The primary benefit of public transparency is not necessarily that errors in analytic results will be detected, although error correction is clearly valuable. The more important benefit of transparency is that the public will be able to assess how much an agency's analytic result hinges on the specific analytic choices made by the agency. Concreteness about analytic choices allows, for example, the implications of alternative technical choices to be readily assessed. This type of sensitivity analysis is widely regarded as an essential feature of high-quality analysis, yet sensitivity analysis cannot be undertaken by outside parties unless a high degree of transparency is achieved.” (OMB, 2001)

Any model is necessarily a simplification of reality, so reviewers can almost always point to simplifications and omissions. The important question is whether these materially affect the conclusions. If reviewers can perform sensitivity analysis themselves, they may be able to shorten their list of criticisms to focus on those that could be material. This could speed things up by focussing discussion on matters of possible importance — reducing the tendency of the review process to bog down on matters that turn out to be of marginal relevance.

Detecting and correcting errors

Errors may be more prevalent in policy models than is generally recognized. Recent audits of operational spreadsheets in government and business find that 50 to 90% of them contain serious errors with respect to the intentions of their authors. See Panko (2000) for a review and (Panko & Sprague, 1996). Moreover, spreadsheet authors and users being typically unaware of these findings are highly overconfident about their reliability. Naturally, few model authors wish to publicize errors. However, there are some interesting lists of news stories of disasters due to errors in spreadsheets collected by the European Spreadsheet Risks Interest Group (EUSPRIG).

Part of the problem is that many spreadsheet errors are easy to make and hard to detect. Formulas using meaningless cell references are much harder to understand and verify than formulas using meaningful variable names. Often there is no way to check the results against real world results because the key results are forecasts or otherwise not directly observable. If an error leads to results that are wrong by 25%, it may not be obvious — unlike bugs in conventional software that create obviously wrong behavior or crashes. In such cases, the most common way to verify models is careful auditing of formulas, which is a challenging way to eliminate errors.

Some of these difficulties of detecting quantitative errors extend to other modeling tools, including visual modeling packages and custom software. Visual modeling packages offer better ways to avoid some kinds of errors, for example visual depiction of dependency graphs. Array abstraction, where a single formula is used to express a mathematical relationship between array-valued variables instead of a separate formula for each cell as in spreadsheets, can massively reduce the number of formulas to be written and checked. Nevertheless, despite the best efforts of modellers and reviewers, it seems likely that all kinds of public policy models contain errors far more often than we would wish.

It is interesting and unexpected to many that open-source software often turns out to be significantly higher quality and reliability, with fewer bugs and security holes, than comparable proprietary software. A key reason seems to be that the larger number of reviewers and developers involved in open-source software means that there are more people with more different perspectives looking for bugs and vulnerabilities, and more people available to fix them. Famously, at least among software developers, Eric Steven Raymond wrote

> “Given enough eyeballs, all bugs are shallow."

which he dubbed *Linus's Law*, after Linus Torvalds, the originator of Linux (Raymond, 1996).

Open-source software code also tends to be cleaner and clearer than closed-source software — and so easier to verify, maintain, and extend. This may be because it is written by programmers who expect their code to be read by their peers, as well as because incomprehensible or poorly documented code is more liable to be cleaned up or replaced by someone else. It seems reasonable to hope for the same benefits of fewer errors and faster detection and fixing of errors for open-source policy models.

Confidentiality and intellectual property

OMB recognizes a major limitation to transparency and reproducibility:

> “Public access to original data is necessary to satisfy this standard, though such access should respect confidentiality and other compelling considerations.” (OMB, 2001)

In many cases, it is possible to aggregate detailed confidential data to preserve anonymity without losing much transparency, since most of the model works on the aggregate data. As OMB suggests, if it is important to review material containing unaggregated confidential data, e.g. to review the aggregation methods, it is often possible for reviewers to work under a nondisclosure agreement (NDA). Obviously, this prevents general transparency of open source for that element of the model.

Another “compelling consideration” is the intellectual property of model authors. It is not uncommon for regulatory agencies to use results from proprietary models created by consultants. In such cases, model owners are understandably reluctant to release the model source code. Sometimes they may be willing to restrict release under a nondisclosure agreement to reviewers approved by the model owners as noncompetitive. But, that is not very satisfactory since it precludes direct access by most reviewers, such as stakeholders, industry, or community groups.

> "In situations where public access to data and methods will not occur due to other compelling interests, agencies shall apply especially rigorous robustness checks to analytic results and document what checks were undertaken. Agency guidelines shall, however, in all cases, require a disclosure of the specific data sources that have been.” (OMB, 2001)

The bottom line is that full transparency and reproducibility of analysis is incompatible with the use of proprietary models. One response would be for agencies to adopt as policy that they will not use proprietary models as the basis for public policy. For areas where no open-source models are currently available, either model proprietors would choose to release the source for their existing models, or other modelers would be commissioned to create new open-source alternatives.

Open source does not guarantee transparency

Publishing software code or a policy model as open source does not guarantee its transparency. If the code is poorly documented, uses incomprehensible variable names (such as cell references in spreadsheets instead of meaningful names), or “spaghetti code”, it may be hard or impossible to understand it. Making code clear takes considerable effort and skill. Ideally, all policy modelers would do this anyway, to enable easier verification, maintainance, and extension of models. When a public agency engages a consultant to build a policy model, one would also hope they would insist on clear documentation. Sadly, with the exigencies of short deadlines, changing model objectives, and inexperienced modellers, this is not always the case.

There exist guidelines for improving the transparency of spreadsheets, such as: Separating of inputs and internal computations, clear organization, consistent documentation, use of names instead of cell references, and avoiding unduly complex formulas. (e.g. Powell & Baker, 2003; Raffenberger, 2000) Unfortunately, awareness of these guidelines is spotty among policy analysts (and most spreadsheet builders), and they are usually ignored.

Spreadsheets are not well suited for creating larger models, because of their limited support for modularity and for managing arrays, especially with more than two dimensions. They do not make it easy to modify dimensions, such as extending the time horizon, adding scenarios, or other dimensions. These features also inhibit reusability and extensibility of spreadsheets — important for evolving a family of models and creating a larger community of collaborating modellers.

Visual modeling tools, such as Analytica, Stella/iThink, and Extend, use influence diagrams or systems diagrams to depict variables and their relationships as nodes and arrows. These diagrams substantially assist transparency offering a higher level representation similar to an expert modeller's mental models of the problem. These diagrams constitute "live documentation" that is guaranteed to be consistent with the underlying mathematical relationships — unlike flow diagrams in conventional software documentation that can easily become inconsistent as a model is modified independent of the documentation. Perhaps more important, the diagrams are used in designing and implementing the model, so they encourage clear thinking and communication about model structure from the beginning, rather than conventional documentation which is often written by someone other than the lead model designer as a final task.

There is room for considerable improvement in model transparency and clarity of documentation for models of all kinds — using spreadsheets, visual modeling tools, or custom software. It might be helpful for an organization like OMB to develop guidelines, and point to outstanding examples of transparency. But, a commitment to open-source policy modeling is itself likely to have substantial benefits towards the goal of transparency. Modellers who know their work will unavoidably be exposed to public scrutiny, especially by their peers, will have strong incentives to make sure their models are clearly structured and documented. And if they do not, other modelers can clean them up, document them, or just replace them. Either way, model transparency should improve over time.

Communities of model users and developers

Given the complexity, the large stakes, and hence political and scientific controversies involved, will open-source policy models lead to clearer and more comprehensive models — or will they degenerate into incoherent messes, with incompatible model elements, and perhaps even sabotage by extremists? If we as a society are to base important policy decisions on the results of such models, we need to be sure that we understand all elements of the model, and who was responsible for them. Policy modelers can gain inspiration from the open-source software development, as well as large-scale open collaborations, such as Wikipedia. We can see similarities, but also important differences in these enterprises.

Prior to the success of open-source projects, it was a truism that more programmers do not necessarily lead to faster development or better quality software. It was surprising even to programmers that open-source projects could be so successful with so many contributors, and little formal structure or project management (Raymond, 1996). It is also amazing, especially to publishers of proprietary encyclopedias, that Wikipedia turns out to be so accurate, given that literally anyone can contribute and edit the text — with only modestly higher error rates than Encyclopedia Britannica and vastly greater coverage according to a recent study ().

Wikipedia, as well as open-source software, demonstrate the remarkable efforts and creativity that some people are willing to contribute unpaid to a collaborative project that really makes a difference in the world. It seems plausible that substantial communities of contributors to policy modeling could grow around particular policy issues.

It turns out that the internal organization of open-source projects is not as unstructured as one might think, albeit the structure is often informal and adhoc. The best-known open-source software products, such as Linux, Apache, My SQL, and Firefox have millions of users. Anyone can download the source code, but the number that actually do so is a tiny fraction. Hundreds are involved in finding and fixing bugs. Typically tens or fewer are in the core team, who make large contributions and decide which changes do or do not go into the next release. Often there is one person — most famously Linus Torvalds for Linux — who acts as a benevolent dictator in recruiting and orchestrating the core team.

Wikipedia has hundreds of thousands of contributors, and a much smaller community of editors who are focused on what changes and how it is organized. While anyone can add or change text, its technology makes it easy to back out edits that editors do not deem valuable.The advantage of the huge number of contributors is that there are many people (eyeballs) reviewing material, who can elect to be automatically notified of changes, and who are likely to spot and remove undesirable additions quickly. This makes it less appealing to would-be vandals. Again, one benevolent dictator, Jimmy Wales the originator of Wikipedia, helps resolve disputes and develop rules for editors to make for smooth development.

By comparison, policy models typically have few “end users” who actually run them — sometimes a handful, up to a few tens. However, policy models, when they affect policy decisions, can affect millions of people and billions of dollars of costs and benefits. Hence, there are often many people and organizations with strong interests in a model, concerned about its assumptions, reliability, objectivity, and conclusions. Such organizations include the government agencies, groups at universities, industry groups, environmental and social justice organizations, and community groups, as well as individual citizens.

There may be only be modest numbers of people with the expertise to review and critique a policy model down to the source code — still fewer with the skills to build one. Some of these organizations have members with the requisite expertise, and others can hire them. Still others may find people with sufficient interest to contribute their skills gratis. As the open-source software projects have demonstrated, it does not take a huge number for a successful project. But, it does appear to take enough continuity to form a community of people who know each other, at least by reputation, email, and webconference.

Controversy and duelling models

One major difference between policy modeling and existing open-source software or Wikipedia, is the degree of controversy about policies, which often extends to the models and science that inform them. The functional goals of conventional software are usually reasonably clear. If programmers have different ideas about how best to reach them, they can write code to test and demonstrate their ideas, and find out what users find most useful. Occasionally, when there are deep differences, open-source projects have been known fork into two or more versions with their attendant communities. Ultimately, the community of developers and the market of users determine which version succeeds, and which falls by the wayside. Wikipedia, like conventional encyclopedias, is constitutionally focused on “accepted” knowledge. Articles are supposed to avoid topics with a point of view for which there is not a consensus.

For policy models, user and developer communities with opposing views on a policy are liable to develop divergent models reflecting their different views. (To a limited extent, that already happens in some areas.) Forking may be the rule rather than the exception. It remains to be seen how this plays out as communities develop around open-source policy modeling. It may prove hard to develop a single policy model relating to a controversial issue -- so, two or more communities may emerge creating duelling models. Such counter-modeling can sometimes be informative and productive.

With policy models, unlike conventional software, it is often possible to combine duelling models as different versions or scenarios within a single meta-model that comprises a broader set of possible assumptions. This approach makes it easier to compare and debate combinations of assumptions and their implications. Software capable of representing multiple alternatives or scenarios, along with pedigree management to identify who is responsible for the various elements, can greatly facilitate this approach.

Patches and pedigrees

If public agencies are to base major policy decisions on model results, they need to be sure that the model is sound and know who has made what contributions to it. It might seem that an open-source model would preclude that. However, even existing open-source software licenses can include a mechanism to track who has made what contributions (pedigree), allowing one to select versions with or without particular contributions.

A major incentive for contributors to open-source projects including Wikipedia is that contributions are public and visible to their peers. Projects vary in the granularity of tracking who is responsible for which improvements. Even though open source licenses allow anyone to modify and redistribute the code, they contain a provision that enables careful tracking of who did what. According to the Open Source Definition, used as the basis of many open source licenses:

> “Encouraging lots of improvement is a good thing, but users have a right to know who is responsible for the software they are using. Authors and maintainers have reciprocal right to know what they're being asked to support and protect their reputations.” (Open Source, 2000)

This is the rationale for section 4 of the Open Source Definition: *Integrity of The Author's Source Code*: An open-source license may specify that you cannot modify source code directly. Instead, you must package your modifications as a separate *patch* that changes or extends the functionality of the original code. This patch mechanism maintains clarity about the source of any differences between the original code and the modifications, labelling the original code and each patch with its author.

When there are many contributors and multiple versions, tracking and managing them can be quite challenging. Interactive development environments (IDEs) widely used for creating software and version management tools can facilitate the process. This would be especially valuable for policy models where responsibility and control of hierarchical version trees is critical. It may be necessary to extend such software with what we might call a *pedigree manager* to enable a reviewer to select any version from the tree, and to see who was responsible for each part, and who else has reviewed, critiqued, or approved it.

Organizational Perspectives

It is hard to predict exactly how a public commitment to open source will change the policy modeling process. Over time, open-source policy modeling has the potential to transform the way policy models are developed, and the way they influence public policy — the ways we arrive at those important decisions that we need to make as a society. The lessons from open-source software and open collaborative projects such as Wikipedia suggest the ripple effects could be profound. There are likely to be significant benefits and challenges for each of the types of organization involved:

For public agencies and regulatory bodies that perform and commission policy models, it has some clear benefits: It enables review by peers and stakeholders to be more comprehensive. The ability for reviewers to perform sensitivity analysis means that their comments may be better focused on substantive issues, with fewer "trivial" criticisms. Model bugs can be found and fixed fasterIt ends the danger of agencies becoming "captive" to consultants with proprietary models. Greater model reusability and extensibility could make model development faster and cheaper.

Making regulatory analyses more transparent can, of course, also make life more complicated for government policy makers and staff, as they have to deal with a wider range of comments and critiques. OMB, however, has already committed them to that.

For stakeholder groups, including industry associations, environmental and social justice groups, and community groups, it removes a crucial limit to the depth of their review. These groups can perform their own sensitivity analysis to critical assumptions. They can even extend a model to address ommitted issues that they regard as important. They can become more active participants in the modeling process. Some industry groups and a few NGOs already have staff with the modeling expertise to do this, but many do not. The development of a wider community of modelers with expertise in particular models, or model types, will make such experts more easily available as consultants to such organizations, paid or pro bono.

At universities and non-profit think-tanks, some policy modelers already publish their models as open source. As a source of modeling expertise and new techniques, as well as trainers of modelers, these organizations have much to contribute. If the models used by public agencies to support policy making and regulation are also open source, it makes it easier for universities and think-tanks to be more intimately involved in reviewing and contributing to these models. This expands the community of expertise, improves the quality of the models, and could increase the chance that their contributions will have real influence on policy decisions.

Consulting firms with proprietary models may find this proposal disturbing. As a sometime consultant for regulatory policy analysis, software designer and entrepreneur, who has invested substantially in developing proprietary software, I can sympathize. Some may be tempted to fight this proposal or delay its adoption. I suspect that others will find greater success in embracing it, and demonstrating their ability to create transparent, clear, and extensible models to their clients. Government could encourage firms to release existing proprietary models as open source by indemnifying them for any errors in previous analyses that may become apparent after release.

Open-source policy modeling is not going to eliminate the need for consultants to create, extend, and apply these models — just as open-source software has created major opportunities for software developers with expertise in open-source products to extend and adapt them for the needs of industry and government. Programmers who have made major contributions to creating high-quality open-source software often find lucrative contracts and jobs — as well as the satisfaction of recognition among ones peers. The same can happen for effective open-source policy modelers.

As a citizen, it seems to me that open-source policy models offers compelling advantages for expanding the notion of transparency. It enables policy making to benefit from a wider community of reviewers, stakeholders, and contributors to models. It reduces the chance that policy making will be captured by a particular interest group to the exclusion of others. It offers the possibility of increasing the quality of policy models while reducing the expense of developing them.

References

Economist 2006: "Encyclopaedia Britannica takes on Nature", *The Economist*, Mar 30th 2006.

EUSPRIG: *Spreadsheet mistakes - news stories*. European Spreadsheet Risks Interest Group. http://www.eusprig.org/stories.htm

Henrion, 1984: "Computer aids for a dialectical approach to designing policy models", M. Henrion, in *Design Policy: Design and Information Technology,* R. Langdon & G. Mallen (Eds.), The Design Council, London, 1984, pp. 53-61.

Henrion & Morgan 1985: "A Computer Aid for Policy and Risk Analysis", M. Henrion & M. G. Morgan, *Risk Analysis*, Vol 5, No 3, 1985, pp. 195-208.

Lumina, 2005: Analytica User Guide: Release 3.1 for Windows, Lumina Decision Systems, Inc. http://www.lumina.com

Morgan & Henrion, 1990: *Uncertainty: A Guide to Dealing with Uncertainty in Quantitative Risk and Policy Analysis*, M. Granger Morgan and Max Henrion, Cambridge University Press, New York, 1990.

OMB, 2001: *Data Quality Guidelines:* Guidelines for Ensuring and Maximizing the Quality, Objectivity, Utility, and Integrity of Information Disseminated by Federal Agencies. Office of Management and Budget. September 28, 2001.
http://www.whitehouse.gov/omb/fedreg/reproducible.html

OMB, 2003: *Circular A4* Sep 17, 2003, Office of Management and Budget. Guidance to assist analysts in the regulatory agencies by defining good regulatory analysis.
http://www.whitehouse.gov/omb/circulars/a004/a-4.html

OMB 2004: *Information Quality Bulletin for Peer Review*
http://www.whitehouse.gov/omb/inforeg/peer2004/peer_bulletin.pdf

OMB, 2006: *Proposed Risk Assessment Bulletin.*
http://www.whitehouse.gov/omb/inforeg/proposed_risk_assessment_bulletin_010906.pdf
Draft released Jan 2006, final comments due June 15, 2006.

Open Source, 2000: The Open Source Definition. Open Source Institute.
http://www.opensource.org/docs/definition.php

Panko, 2000: *What We Know About Spreadsheet Errors*, Raymond Panko, 2000.
http://www.lumina.com/ss1/SpreadsheetErrors.htm

Panko & Sprague, 1996: Panko, Raymond R., and Sprague, Jr., Ralph H., “Hitting the wall: errors in developing and code inspecting a ‘simple’ spreadsheet model,” *Decision Support Systems*, Apr. 1998; 22(4): 337-353.

Powell & Baker, 2003: *The Art of Modeling with Spreadsheets: Management Science, Spreadsheet Engineering, and Modeling Craft*, Stephen G. Powell, Kenneth R. Baker, Wiley, 2003.

Raffenberger, 2000: *The New Guidelines for Writing Spreadsheets* John F. Raffensperger, http://www.mang.canterbury.ac.nz/people/jfraffen/spreadsheets/index.html

Raymond, 1996: *The Cathedral and the Bazaar*, Eric Steven Raymond
http://www.catb.org/~esr/writings/cathedral-bazaar/cathedral-bazaar/index.html