Authors' response to discussion of *The relevance of Open Source to Hydroinformatics* by Hamish Harvey and Dawei Han, 2002 *J. Hydroinf.* 4(4), 219–234

Hamish Harvey and Dawei Han

Professor Abbott presents a great deal of stimulating thought on the extension of lessons learned in the field of Open Source software development to hydroinformatics generally, rather than just the application of those techniques to hydroinformatics software development, while Professor Cunge raises some specific and challenging questions regarding the conclusions of our paper.

The distinction between endostructure and exostructure, though with different terminology, is precisely the subject of Brown & Duguid (2000), who observe the pitfalls of *imposing* structure on knowledge organisations. Levine *et al*. (2000), with their 'markets are conversations' mantra, go a stage further in suggesting that companies which fail to allow an endostructure to emerge will suffer in competition with those which do.

There is at least some validity in considering exostructure and endostructure to be structures of power and influence, respectively. Influence is surely more rewarding than power, a point apparently not understood by legions of politicians and bureaucrats. Of course it is harder to acquire and hold influence, since it must be earned, not bought or taken. Progression up the echelons of power almost invariably involves making extensive use of personal networks of communication and influence, and exostructure can inhibit, but never fully suppress, endostructure.

These dual structures appear outside the commercial arena, too, from street gangs to Enron, from local voluntary projects to national and international government. It seems probable that the generally dysfunctional nature of hippy collectives results from blocking the emergence of an endostructure in eagerness to avoid establishing an exostructure—the baby is thrown out with the bathwater.

The issue of the takeover of academe by bureaucrats is an interesting one. Perhaps an even more extreme example is that of commercial academic publishing. Here we have a system dedicated to communication and to the establishment of structures of influence and reputation, subverted over the years in the interests of financial profit. The separation of those who benefit from publication and thus the services provided by publishers (the authors) and those who pay for these services (the readers, and even then only indirectly through libraries from overheads) results in the total absence of any kind of competitive pressure on publishers to provide services of value. Considerable debate on this issue has taken place, some of it notably on the web site of the prestigious journal *Nature*[1].

If endostructure is emergent, it is also dynamic. This suits it well to the software world, where change happens on 'Internet time', and in a truly competitive environment the organisation unable to cope with that change is doomed. Thus we see large exostructure-bound corporations fighting rear-guard actions through the courts or out of them—with software patents or monopoly positions—to protect themselves from eclipse. These manoeuvres, legal and illegal, might be seen as the exercise of Abbott's 'naked physical force'—the only option open, since the endostructure-based organisations are able to make optimum use of the Internet and have a huge lead in communication. As Levine *et al*. (2000) suggest, the endostructure

[1] http://www.nature.com/nature/debates/e-access/

of a company extends well beyond the walls of the company's offices to include its customers and even competitors.

These are all, as well as lessons to be learned from Open Source, possible reasons why the Open Source model is so effective. In Open Source projects the *only* structure is endostructure. Open Source projects do not waste valuable time and money acquiring (and possibly defending) patents, for example. The boundary between developer and user is blurred, with users contributing bug reports and sometimes fixes. Communication flows freely and influence is earned (and occasionally lost).

There are two related issues embedded in Abbott's discussion which demand closer attention. These are in the postmodern 'consumers of knowledge' notion, and in the assertion that the present environment necessitates that we sell (or sell the use of) encapsulated knowledge. The second of these is also at the heart of Cunge's objections and is treated later.

Is the change from knowers to consumers of knowledge something the postmodernists intended to *draw attention to*, or to *establish as a goal*? Certainly, it might be seen as something to be approached with extreme caution, as the provision of consumable knowledge can be expected to reinforce existing power relationships to the benefit of the knowledge-having elite (in which the members of this journal's readership might reasonably be numbered) at the expense of a knowledge-consuming underclass.

Trade of money for goods with the 'third' world is surely in its essence unsustainable. A third world country buying knowledge (in one-time use or reusable encapsulated form) from Europe or the United States, for example, involves third world money paying first world salaries and overheads. In order to avoid haemorrhaging money, the third world country must export goods of equal cost, which leads to such evils as the growing of cash crops for export in areas of desperate food shortage. At the same time, covering the cost of supplying such encapsulated knowledge from overseas development budgets is risky. Aid breeds dependence, and the provision of packaged knowledge (to the 'third' world in particular, but not exclusively) is as risky as the provision of food, or the unthinking installation of 'first' world technology. A third option—of no interaction whatsoever—can be dismissed as absurd.

The point here is not to dismiss the advantages—and increasingly the necessity—of encapsulation as a tool for managing complexity. It is no new insight that the technical use of that word in software engineering is nothing more or less than a specific example of knowledge encapsulation. We could think of encapsulating knowledge in a box with a loosely fitting lid—the contents will not spill out by accident, nor do they distract the user, but the person who would or must have a deeper understanding of the encapsulated knowledge can take the lid off.

The loose-lidded box effect is not created by the simple availability of source code to software, as the knowledge in source code form is merely *less* obfuscated than in binary form. More modular approaches to software construction—and in particular the ability to compose models from components at run time, progressively encapsulating knowledge—are essential to make the thus-encapsulated knowledge useful. There is no room to expand on these notions here: they are covered in more detail in Harvey (2000).

There is an urgent need to cultivate balanced forms of exchange between the first and third worlds which, ideally, help in slowly bringing the world into better balance. Nowhere is this more important, of course, than in the area of knowledge regarding water and therefore in its representation in software. It is imperative that we *find* ways to work together and to build local expertise, and one opportunity for doing this is to finance software development by some means other than treating software as a product. This would enable education leading to cooperative work between 'worlds' without commercial loss. We therefore assert—while postponing the discussion of economic viability—that it is at least *desirable*, in terms of our social responsibilities which result from our enhanced duty of care as experts in water and software, that we establish an economic structure such as that offered by the Open Source model.

Let us move onto the questions raised by Cunge. First of all, we acknowledge the criticism that the paper made no effort to study the lessons of the fusion of public domain and commercial in United States government funded software. This would indeed be an interesting and

valuable study, and although time and space do not permit such an exploration here a few points are worth noting. The degree to which software produced by US governmental agencies is free (as in beer) and open varies. Although the basic version of HEC-RAS is available at no cost, its developers discourage distribution or modification of the source code[2], so it cannot be considered open. FLDWAV, on the other hand, appears[3] to be supplied with source code but comes at a cost. Other such software is placed in the public domain (where copyright is surrendered wholly) as distinct from being Open Source software (where copyright law is used to grant specific rights to software users). Public domain software has, in general, failed to attract the interest of developer communities on the same scale as Open Source.

The availability of source code under a licence which allows people to modify the software and distribute modified versions is necessary but not sufficient to stimulate the emergence of a community of contributors. The core developers must generally be responsive to the submission of patches, incorporating them in the central code base or providing good reasons for their rejection. They must participate in online discussions of the software. They must keep the community up to date with their plans for the software, with feature plans and release schedules on the web. Giving away software at no cost, with or without source code, removes the supplier's ability to generate revenue from the software without enabling any means of obtaining value from giving it away by alternative means.

As regards the problems which arise from the inability of end-users to accept that hydroinformatics tools, in fact, cannot be used as black boxes, we find it hard to see how proprietary and Open Source software differ in this regard. The issue of inflated expectations of what software can do for us without any personal intellectual input is a deep problem, and must be addressed through education and possibly in the design of software and graphical interfaces. Similarly the problem of software containing bugs (or being wrong—the distinction is not clear) is a general one. In fact, the general liability disclaimer which accompanies all software—whether proprietary or Open Source—is usually more visible in Open Source software.

This leads logically to the issue of 'responsibility for content'. The assertion that, in Open Source, this is nil is based on a conflation of *trust* with commercial interest. To be sure, a company selling software has an interest in their users and potential users trusting their software, but it is by no means the case that this model is the only way of establishing trust, and the observation that users have *excessive* trust in software suggests that this conflation is already problematic. The strong emergent endostructure of Open Source projects highlighted by Abbott is, as noted above, a structure of influence, and influence is earned by establishing trust.

The level of user community involvement—including users who do not take an active part in development—in Open Source projects leaves such trust structures visible to the world. Half an hour browsing the mailing list archives of an active Open Source project will provide interested parties with a good impression of where influence lies, and since influence can only be earned, this can be expected to be an indicator of trust.

This structure of trust operates from the level of individual developers up to that of projects. With proprietary software products, where communication between user and supplier and, critically, between user and user is limited, the emergence of such trust relationships is restricted. It is not propaganda regarding the availability of 'anything' on the Internet which leads to problems of misplaced expectations, but the general lack of visible trust structures, and a move towards the Open Source model of software production offers an *opportunity* to address, rather than exacerbate, this problem.

As observed in the paper, issues surrounding the distinction between trust in the quality of the software itself (in terms of bugs causing program crashes, for example) and trust in the scientific or engineering content of the software remain to be explored. The academic tradition of establishing trust by independent verification is very much hindered by the closed nature of proprietary software in a way which it is not in an Open Source environment, even if the technical and social mechanisms by which this might be achieved in an Open Source environment await

[2]http://www.hec.usace.army.mil/software/software_distribution_policy.html
[3]http://www.ntis.gov/search/product.asp?ABBR = PB2003500004

exploration. Again this suggests that Open Source provides an opportunity to rectify an existing problem rather than introduce a new one.

This brings us finally to the suggestion by both correspondents that the current environment necessitates that we continue to sell software or its use. Since this is closely related to the types of software which are suited to Open Source development, we can discuss these together. First, it should be observed that precisely this belief led to the complacence followed by panic displayed by the proprietary shrink-wrapped software industry at large when faced with the growing Open Source movement.

It is accepted, however, that there are economies of scale at work in most currently successful Open Source projects, and the distinction Cunge makes between operating systems and hydroinformatics software is presumably intended to highlight this. The particular choice of comparison disguises this somewhat—in both the Open Source world and in hydroinformatics a wide range of software types are produced, from operating systems, through server software, to desktop applications in the former, from custom software such as flood forecasting systems to relatively 'mass' market modelling tools in the latter.

The important difference in the case of product style software, such as operating systems and applications (in hydroinformatics, modelling tools), is in the potential base of contributors, which we can assume to be related to the base of users of the software and thus dependent on the size of the market, although it is also related to the nature of that base of users (are they technically adept, for example?).

As observed in the paper, a key characteristic of successful Open Source software projects is the high modularity of the software architecture. If modelling tools could be modularised to the extent that modelling engines, model implementations and graphical interfaces could be developed substantially independently, a picture emerges of the modelling engine—now only a framework into which model implementations are hooked—as *infrastructure*. Model implementations are then subject to some real competition since a set of models suited to the task in hand can be selected, and are likely to become commoditised to a degree—in particular, the market is opened to smaller players and technology transfer from academia into practice is greatly facilitated. This does not remove value from the system—it increases it by reducing cost and spreading risk.

The dismissal of decision support systems (DSSs) as a target for the Open Source approach is interesting. DSSs are highly customised applications with common elements and, while the market may at present be small, if it is not to grow rapidly over the coming years then a large proportion of the content of past hydroinformatics conferences will have been wasted effort. DSSs can be expected to have considerable common elements yet always require customisation and extension for a given application. The market for DSSs is at least partly constrained by their cost.

An Open Source approach might allow those who make (or try to make) a business from building DSSs to defray the costs and risks of establishing the frameworks necessary to allow customised DSSs to be built more cheaply. As more DSSs are built, the shared framework can be expanded to include more functions included as custom features in earlier systems. Collaborating on an Open Source framework while selling as a service, the *development effort* required to customise it is the model followed by Zope Corporation[4] with its Zope[5] application server with great success.

The authors thank the correspondents for their discussions, and hope that the issues have been at least adequately enough addressed to continue the discussion. It is understood that Open Source is no silver bullet, but it is also clear that the development of economically viable mechanisms for funding software development is an important issue. We maintain that the present structure—based on the artificial treatment of software as a product—is flawed, and it would serve our discipline and industry well to begin thinking about and experimenting with ways to address this problem.

[4]http://www.zope.com/
[5]http://www.zope.org/

REFERENCES

Brown, J. S. & Duguid, P. 2000 *The Social Life of Information*. Harvard Business School Press, Boston, MA.

Harvey, D. 2000 A generic modelling framework component for hydroinformatics systems. *PhD thesis*, University of Bristol.

Levine, R., Locke, C., Searls, D. & Weinberger, D. 2000 *The Cluetrain Manifesto: The End of Business as Usual*. Pearson Education, Harlow.