Water Resources Modeling Tools -
Open Source Code versus
Proprietary Software

Edward H. Burgess, William James and Lewis A. Rossman

The water resources engineering community has available an array of computer
models that can be applied to support analysis, planning and design of water
resources systems. Some models are software that reside in the public domain
as open source code; some models are commercial software for which the
source code is the proprietary property of its owner. In selecting a model for
use in supporting their work, water resources engineers are therefore faced
with a basic choice - to use a public domain model, or a model that is commercial
software. There are many factors that must be considered when making this
choice. Careful consideration of the relevant factors reveals that there are many
benefits of open source software that favor the use of public domain models.
In addition to the selection of software on an individual basis, there are also a
number of factors that water resources engineers should consider from the
broad perspective of the benefits to their profession in supporting the
continued advancement of public domain models. The various factors that
should be considered are identified and evaluated.

© CHI 2004 www.chijournal.org ISSN: 2292-6062 (Formerly in Innovative Modeling of Urban
19.1 Introduction

As the engineering software industry continues to evolve and mature, the development of these programs becomes increasingly the domain of professional programmers producing stand-alone commercial products, while fewer practicing engineers write their own programs to solve day-to-day engineering problems. This is probably true in all fields of engineering as well as in other disciplines. Ironically, as engineers use computers more and more in their daily work lives, fewer and fewer develop the expertise to encode solution algorithms for common engineering problems - the growing availability of engineering software reduces the need for practicing engineers to develop this expertise.

Engineers today are faced with a growing array of commercial software tools that can be used to simulate water resources systems. These tools often represent alternatives to the various models that have long been available in the public domain - including those available from the U.S.EPA (e.g. SWMM, HSPF, etc.) and other federal agencies. In some cases the available commercial tools incorporate public domain software as a computational "engine"; in other cases the commercial tools include their own proprietary computational engine. Examples of the former include PCSWMM®, MIKE SWMM®, and HEC-PACK®. Examples of the latter include XP-SWMM®, InfoWorks®, MOUSE® and StormCAD®.

It is important to consider the distinction between the model engine (the software that comprises the computational algorithms) and the interface (the software that controls how the user interacts with the engine to create model input and review model output). Much water resources modeling done today employs commercial software with a modern graphical user interface (GUI); most modelers in North America use software that integrates public domain computational engines, while in Europe there is greater usage of proprietary packages. These two approaches to software development represent a fundamental and important distinction in the available models, and one that must be carefully considered in choosing commercial software tools for water resources modeling.

This chapter compares the characteristics of modeling tools using open source as opposed to proprietary computational engines, in the following areas:

- application reliability,
- updates and enhancements,
- code testing, quality assurance/control (QA/QC) and verification,
- modification / customization, and
- stewardship / sustainability.
19.2 Evolution of Water Resources Software

It should be stated here at the outset that this chapter is not a scholarly presentation of research results, but rather a set of both observations and opinions on the subject of software used by water resources engineers. The purpose of this chapter is to provide modelers with a comparative basis to evaluate the available modeling tools and select those that best support their needs. It is hoped that in presenting the various comparisons, it will stimulate thought and discussion in the water resources community about the virtues and risks associated with the available open source and proprietary computational tools.

19.2 Evolution of Water Resources Software

Knowledge of the evolution of water resources software is useful in understanding the open source versus proprietary debate. This section addresses the evolutionary paths of water resources software development and of ownership/distribution separately, and provides only a brief summary of each from the authors' personal observations. More detailed historical descriptions of software development in the various specific areas of water resources are available elsewhere in the literature, e.g. stormwater modeling software (James, 1993; Overton and Meadows, 1976).

19.2.1 Evolution of software development

In the 1960s engineers wrote programs in FORTRAN (or BASIC) to handle relatively simple, but increasingly complex, engineering analyses. These development efforts took place primarily in academic settings, and eventually migrated to specialty consulting firms and federal government agencies. Typically the software was ad hoc in nature, developed to solve a specific problem, and often was not very portable to other applications. Computer hardware played a role in this situation, as it was often difficult to move programs to other computers due to incompatibilities between the various mainframe computer platforms available at the time.

During the 1970s, computer programming efforts became increasingly ambitious and more broadly useful. SWMM is a good example of this trend (Metcalf and Eddy et al, 1971; Huber and Dickinson, 1988; Roesner et al, 1988; James, 1993); HSPF is another (Johanson et al, 1980). But during this period there was relatively little software development activity taking place in the water resources field outside of academic and government (or government-funded)
projects. Any privately funded software development efforts that were undertaken during this period in the water resources field were generally not for the purpose of producing commercial software products.

In the 1980s the development of water resources software began to change rather dramatically. One of the key drivers for change in this field - as with virtually every field which uses computers - was the rapid improvement and deployment of personal computers, which greatly expanded the access of water resources engineers to computers. With the rapid and dramatic increase in the number of computer users in the water resources field, a commercial market for water resources software was created.

Commercial software became increasingly available in many fields, and water resources was no exception. This software was developed by commercial enterprises seeking to sell software to engineers who wanted to take advantage of these newly available computing resources to solve water resources problems. Software firms began to replace engineers working in academia and government as the typical developers of water resources software.

During the 1980s and 1990s the development of commercial water resources software continued to evolve and mature. While many practicing engineers embraced the newly available computer resources and developed their own software early in this period, this practice receded toward the late 1980s and early 1990s. During this period the entire software industry matured, operating systems became graphical (e.g. Microsoft Windows®) and more complex, and practicing engineers found that software development increasingly became the domain of programming experts. Further, as engineers found that the expanding body of commercially available software increasingly met their needs, the need to develop their own tools diminished.

During the 1990s and into the present decade, the commercial water resources software industry continued to expand, evolve and mature. During this period, the focus of software development shifted from the development of solution algorithms required to perform analysis and design computations to the development of graphical user interfaces and interoperability with databases, geographic information system software and such features and tasks outside the computational solution area. These aspects often became important to the marketing and sales of the commercial software, and began to dominate over the computational aspects, which increasingly have come to be taken for granted by users (i.e. as the model user bases expand, relatively fewer users understand the computational aspects well enough to distinguish the various models on this basis).
19.2 Evolution of Water Resources Software

The current status of the commercial water resources software market has led to a troubling state where engineers, following the marketing efforts of commercial software firms, are beginning to focus their attention on the packaging and user interface aspects of the software they use, while largely ignoring the computational aspects that are critical to the proper analysis and design of water resources systems. This topic is addressed in more detail later in this chapter.

19.2.2 Evolution of Software Ownership and Distribution

The evolution of the ownership and distribution of water resources software has of course followed closely the evolution of its development. In the 1960s, engineering programs were generally problem-specific and/or hardware-specific, as noted above, and therefore had limited commercial value. Software was therefore shared freely during that period, and ownership often did not extend beyond the intellectual property of the author. Typically there was no commercial value, nor distribution other than personal exchange of magnetic tapes or punch cards between professional colleagues.

Ownership and distribution issues became somewhat more complex in the 1970s. Programs broadened in scope and portability, which increased their value. However, in the USA, since the federal government was the primary "developer", or since it funded the development through academic institutions and consultants, it was the owner of much of the water resources software developed during this period. Therefore, most water resources software that existed during this period was typically public property, and it was freely distributed on tape/disk by request.

In the 1980s the ownership and distribution of water resources software became significantly more complex. As microcomputers created the commercial software market, the private sector began to overtake the federal government in software development. Private developers of commercial software retained proprietary ownership rights to the software they developed. Under those rights, the developer has the right to license use of the software to those who purchase the commercial software product and the practice of license-based software distribution emerged.

Along with the practice of license-based distribution came an important new restriction for software users - the proprietary rights of software developers universally blocked user access to the source code. This was done to protect the developer's proprietary interests, specifically to prevent others from unauthorized use of the program code for other commercial products.
And with this new restriction placed on access to the source code, the open source vs. proprietary modeling debate started.

19.3 Impacts of Commercial Software on the Water Resources Community

The evolution of water resources software described in the preceding section has had a profound impact on the water resources profession, and has significantly changed how water resources systems are analyzed, simulated, planned and designed. Many of these changes are positive. Better tools are continuously available to engineers and managers, more complex and useful analysis can be performed, better operational information is provided, and planning and design work can be performed more accurately and efficiently. Not only are the tools better, but they are more widely used. The widespread use of computers in all aspects of professional life, combined with the ever greater availability of water resources software, has led to more water resources engineers using software tools - especially computer models - than ever before.

But there is also a down side to these changes. Perhaps key is the loss of user understanding of the underlying computational processes that are performed by water resources modeling software. This has resulted directly from the development and distribution of proprietary commercial software, which blocks user access to the source code (therefore sometimes referred to as "closed" program code). Since engineers cannot review the source code, they can only assume that the algorithms were properly formulated for the intended application, and that those algorithms can be properly applied to their particular situation in each application.

Engineers have increasingly accepted proprietary (closed) modeling software, as they have become accustomed to using closed code in other aspects of their work. For example, because engineers cannot inspect the source code for Excel®, Word®, Outlook® or other software that they use daily, the inability to inspect the source code for modeling software unfortunately becomes increasingly accepted as normal software practice.

To fully understand the importance of this issue, we must first define what constitutes open source software, describe the members of the water resources community who affect and are affected by this dilemma, the nature of the real distinction between open source and proprietary software, and how this distinction has become important to the water resources community.
19.3 Impacts of Commercial Software on the Water Community

19.3.1 Definition of Open Source Software

*Open source* is a term for software that is openly distributed, often under conditions specified in a licensing agreement. Two critical characteristics are:

- users are given access to the source code, which allows them to modify, study or augment the software's functionality; and
- any licensing agreement allows distribution of the initial software and redistribution of that software in a modified form. If users make changes to the software, they may submit them to the community of developers for possible inclusion in future versions (Trimble, 2000).

One important subset of open source software is public domain software. A public domain program is one upon which the author has deliberately surrendered his copyright rights. Public domain software is unlicensed, and no license is required for its use; it is the personal property of the users to use as they see fit (Perens, 1999).

Much of the open source software available in the water resources field is public domain software. The key characteristic of public domain (open source) code is that it can be obtained by anyone, without limitation on the ability to:

- review the source code,
- revise/edit/modify and compile the source code,
- execute the compiled program code, and
- distribute the source or executable code to others.

Open source software should be distinguished from "non-commercial" software or "freeware", which is software for which only the executable code is distributed at no cost. Although open source software is also non-commercial (since the executable program is distributed at no cost), its source code is freely accessible, which may not be the case for freeware.

19.3.2 Water Resources Community - Software Developers and Users

In the context of water resources modeling software, the water resources community can be thought of in terms of two subsets: (i) software developers, and (ii) software users. These subsets are not mutually exclusive, but the members of each subset can be identified. It is useful to do so to understand both sides of the debate among both developers and users regarding open source versus proprietary models.
The software development community is composed primarily of:
- government agencies,
- non-profit research institutes,
- private for-profit software companies,
- consulting engineers,
- academic institutions, and
- public water resource management/utility agencies.

In some cases members of one group work cooperatively with those of other groups, or contract software development to members of another group. For example, consulting engineers may develop software for public agency clients during modeling projects. Or U.S. government agencies may contract out software development projects to consulting engineers and academic institutions (this is how SWMM was developed).

Generally software developed by or for U.S. government agencies is open source, software developed by other public agencies is often non-commercial but not freely distributed or open source, and software developed by the other groups is usually proprietary. However, there are many exceptions to these general characterizations.

The modeling software user community is much larger but limited, for the most part, to three groups:
- water management agencies (at all levels of government)
- consultants (working on behalf of water management agencies)
- academic institutions

All three groups use both open source and proprietary modeling software.

19.3.3 The Open Source vs. Proprietary Distinction

The nature of the distinction between open source and proprietary models has evolved rather dramatically since the advent of water resources modeling software in the 1960s. Since that time, the architecture of water resources models has evolved along two parallel, but not concurrent paths, which eventually began to merge, as depicted in Figure 19.1. The chronology of this evolution can be summarized as follows:

1960s and 1970s: All software was open source - there was no distinction.

1980s: Proprietary modeling software is developed and marketed, which gives rise to two distinct approaches to modeling
software ownership and distribution: (i) commercial software for which ownership is retained by the developer and which is licensed to users; and (ii) public domain software which continued to be freely distributed as open source software. The distinction between open source and proprietary models emerged.

1990s: Proprietary interfaces to public domain computational engines became popular. In this case the interface is typically distributed as commercial software, while the computational engine continues to be freely distributed as open source software. The term interface is used generally here to also cover shells and decision support systems.

19.3.4 The Current Status of Water Resources Software

The water resources modeling software in most common use today tends to generally fall into two variations of commercial software that are described above - either (i) an open source computational engine bundled with a
proprietary interface into a commercial product, or (ii) fully commercial software, with both the computational and interface components distributed as closed proprietary code. It is interesting that the former approach - a hybrid product with a commercial interface and an open source computational engine - tends to be the dominant approach in North America, although the fully commercial models are heavily marketed and are gaining modest user bases. In Europe, however, the fully commercial models are dominant.

Fully open public domain models (meaning both computational engine and user interface) are still available and in use, although in some fields (e.g. sewer system modeling) generally used today for only relatively small or simple applications, or in academic settings. Most users who apply models for large or complex applications find that the increased capabilities and efficiency provided by the various commercial interfaces or commercial models is beneficial or even required. The need for these commercial products is largely the result of limited efforts by many U.S. government agencies over the past dozen years to update their legacy modeling software and add graphical user interfaces to them, which in turn encouraged the development of the commercial modeling products.

However, since graphical user interfaces have today become a virtual requirement for software of any type to be useful, there has been a modest reversal of this trend. The current versions of SWMM5 (Rossman et al, 2003) and EPANET, which are both fully open source software bundled with very functional open source user interfaces, are examples of the current approach being taken by the U.S.EPA. As a matter of broad federal policy, the U.S. government has renewed its support for the development of open source software in recent years (Dorobek, 2000).

To illustrate the current status of water resource software, several examples of each basic approach are listed below.

**Public Domain (Open Source) Modeling Software**
- SWMM4 and 5
- HEC-RAS
- WASP

**Open Source Computational Engines**
- SWMM4 and 5
- HEC-RAS

Used with proprietary model interface products:
- PCSWMM
- MIKE SWMM
- HEC-PACK
19.4 Impact of Software Ownership on Software Selection Decisions

**Fully Proprietary Modeling Products**

- XP-SWMM
- MOUSE
- InfoWorks
- MIKE-11
- StormCAD

There are also several examples of water resources modeling software that do not fit easily into the above categories. These include commercial modeling software that is distributed with a proprietary version of a public domain computational engine (e.g. XP-SWMM), and modeling software built to operate within a proprietary GIS software environment - such as the U.S.EPA BASINS software, which is a non-commercial software package that includes public domain computational engines linked to a commercial GIS software product (ArcView®).

19.4 Impact of Software Ownership on Software Selection Decisions

For the individual, software ownership considerations have a significant impact on the selection of the most appropriate software for solving real world problems. On a broader scale, ownership considerations have important impacts on the role of software in the water resources profession. The specific impacts are discussed in the next six sections, under the following subheadings:

- application reliability
- modification - customization
- updates and enhancements
- code testing - QA/QC - verification
- stewardship - sustainability - support
- other factors

19.4.1 Application Reliability

This is perhaps the single most important factor that must be considered, and it can be argued that this factor alone overrides all others. The engineer must consider very carefully whether or not he/she can affirm the modeling results...
that are produced by software for which the solution algorithms and source code cannot be inspected.

It is sometimes argued that users of commercial spreadsheet software (e.g. Excel®) cannot inspect that source code, and this is not considered an unacceptable limitation by contemporary engineers. Therefore, by extension, it is argued that the same standard should be applied to modeling software. But this is not a valid argument. Spreadsheet software simply executes standard mathematical operations, all of which can be checked manually, as can the formulas “coded” into the cells of the spreadsheet. Modeling simulations on the other hand are complex, computationally intensive operations that cannot in any practical sense be checked by manual methods. Thus the only practical way to confirm the validity of most modern-day modeling results is to confirm the algorithms used and the proper coding of those algorithms by inspection of the source code (in addition to confirming the proper representation of the physical system and proper model execution).

A significant benefit of open source models is realized during the output debugging process in many practical applications. Often when model output produces unexpected results, examination of the source code provides useful insights into the problem, such as revealing situations where the modeled representation is inconsistent with the program code. This is of course not possible with proprietary models.

19.4.2 Modification - Customization

<table>
<thead>
<tr>
<th>Open source</th>
<th>Proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minor changes to accommodate specific applications can easily and quickly be made</td>
<td>Customization is generally not feasible:</td>
</tr>
<tr>
<td>Code must therefore anticipate every possible user need</td>
<td></td>
</tr>
</tbody>
</table>
19.4 Impact of Software Ownership on Software Selection Decisions

It is often helpful or even necessary to adjust program code to accommodate existing database characteristics (e.g. to read long record identifier strings, etc.) and to properly simulate unique physical system conditions or features. These changes can be made quickly and easily by a user knowledgeable in the appropriate programming language and with access to the necessary compiler software, if the program code is open. But with proprietary models such changes are much more difficult to accomplish, for several reasons:

1. the software developer/owner must make the changes - and may be unwilling or unable to do so;
2. the developer/owner may have a regular (often annual) release schedule, and even if willing and able to make the change, may defer the code changes and/or release of the updated software until the next regularly scheduled release; and/or
3. it may be difficult to fully and properly communicate the needed change to the developer, and it is difficult to confirm that the change was correctly made, since the revised code cannot be inspected by the user.

Further, if the engineer cannot inspect the source code, it may be difficult to even determine that a modification to the code is required to accommodate the prototype system.

Also, since proprietary software can be difficult to customize, users may be forced to develop workarounds or approximations in the modeled representation of the system to allow it to accommodate the software, rather than the software accommodating the modeled representation of the system.

19.4.3 Updates and Enhancements

<table>
<thead>
<tr>
<th>Open source</th>
<th>Proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Updates are irregular (e.g. every 15 years or so in some cases)</td>
<td>Updates are regularly released</td>
</tr>
</tbody>
</table>
| New features can be added at any time by anyone | Code changes are only made by the software developer/owner, who controls:
  * Features to be added and/or modified
  * Schedule for changes |

One positive attribute of proprietary software is that the profit motive associated with the commercial enterprise drives innovation, and typically spurs the regular enhancement, updating and release of new versions of the software. To a large extent a similar process takes place with open source software, as the
need to support project work (also a commercial enterprise) drives considerable enhancement and updating of open source software. And of course one advantage of open source software is the unrestricted ability of users to perform the updates, whereas with proprietary software the specific enhancements to be made and the update schedule are completely controlled by the developer.

19.4.4 Code testing - QA/QC - Verification

<table>
<thead>
<tr>
<th>Open source</th>
<th>Proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Testing</strong>: Individual coders test code enhancements / modifications (not standardized)</td>
<td><strong>Testing</strong>: Strict and controlled testing process can be implemented</td>
</tr>
<tr>
<td><strong>QA/QC</strong>: Variable according to contributor</td>
<td><strong>QA/QC</strong>: Easier to implement standard procedures for QA/QC</td>
</tr>
<tr>
<td><strong>Verification</strong>: Entire user community can review and verify code</td>
<td><strong>Verification</strong>: Only internal development team can review and verify code</td>
</tr>
</tbody>
</table>

Another positive attribute of proprietary software is that the QA/QC process for software testing and verification can be a strictly controlled and standardized operation, since the program code is accessible only to a limited group. In contrast, software QA/QC practices are by definition variable for open source code, according to the practices of the specific programmer making the changes. However, there are several factors that offset any perceived advantage that proprietary software may have in this area. First, the QA/QC practices for proprietary software are by nature a closed process, thus their extent and effectiveness is not known. Second, only the internal development team can review and verify proprietary source code, whereas the entire user community can review and verify open source code.

An important factor in this area is the role of the “caretaker” for open source modeling software. The caretaker role is often handled by the public agency “owner” of the software, which may be supplemented by others in the academic or even consulting communities. Standardized QA/QC practices can be and are implemented by these caretakers.

Another important advantage for public domain models in this area is attributed to the size of the user base. Since public domain models are freely available, they typically have much larger user bases than commercial models. And since a critically important component of model QA/QC is accomplished by stressing the model in applications, a significantly higher level of QA/QC testing can be imposed on public domain models.
19.4 Impact of Software Ownership on Software Selection Decisions

19.4.5 Stewardship - sustainability - support

<table>
<thead>
<tr>
<th>Open source</th>
<th>Proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dependent upon volunteer entity to sustain the model</td>
<td>Proprietary interest exists for sustaining the model</td>
</tr>
<tr>
<td>Stewardship can be diffused and vague</td>
<td>Stewardship is centralized within a clearly-defined owner</td>
</tr>
<tr>
<td>Open-source fosters a large user community to:</td>
<td>Proprietary models tend to have a “customer base” rather than a “user community”</td>
</tr>
<tr>
<td>* assure continued stewardship</td>
<td></td>
</tr>
<tr>
<td>* support the model via user networking</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Support is typically a fee-based service</td>
</tr>
</tbody>
</table>

There are clearly advantages and disadvantages associated with both proprietary and open source models in this area. The profit motive associated with the commercial software enterprise creates a clear incentive to properly sustain the model, and its steward is clearly identifiable. While support of proprietary models is typically a fee-based service, there is a clearly identifiable party with an incentive to provide service to whom users can direct their support needs.

These advantages of proprietary models also pertain to proprietary model interfaces, and have been partly responsible for the popularity of the commercial interface products, as the same parties that support the interface software also provide support to their customers for the computational engines.

Perhaps the greatest advantage for users of open source software in this area is attributable to the intangible yet real benefits of belonging to a large user community, which is fostered by the non-commercial nature of open software. This community in many cases provides a very effective user support network (e.g. the SWMM User Group and its forum for discussion maintained as the SWMM listserver at the University of Guelph).

In contrast, proprietary software tends to have a “customer base”, in which users look to the vendor - rather than to each other - for support. And the vendor will only support the product as long as it is a part of their product line. There have even been instances where a vendor’s ability to sustain and support a product can depend on one or a few key staff, and if those staff are no longer employed by the vendor, product support (and even the product itself) becomes unavailable.
19.4.6 Other Factors

<table>
<thead>
<tr>
<th>Open source</th>
<th>Proprietary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Impact on the profession:</td>
<td>Impact on the profession:</td>
</tr>
<tr>
<td>• “intelligent users”</td>
<td>• “Button pushers”</td>
</tr>
<tr>
<td>• Encourages exchange of ideas</td>
<td>• Proprietary interests can impede idea sharing</td>
</tr>
<tr>
<td>• Links between the academic and practitioner communities</td>
<td>• Little value in academic settings</td>
</tr>
<tr>
<td>• Useful teaching tools</td>
<td></td>
</tr>
<tr>
<td>Cost – core computational code is available at no cost (interfaces: $400-4,000)</td>
<td>Cost – generally significantly more costly than OS models/interfaces</td>
</tr>
<tr>
<td>Market size – open source does not require a revenue stream to sustain a model</td>
<td>Market size – market for specialized tools often insufficient to sustain competitive commercial enterprises</td>
</tr>
<tr>
<td>Ease of use (GUI) – relies on user community to improve</td>
<td>Ease of use (GUI) – greater incentive to improve GUI (increases user acceptance / sales)</td>
</tr>
</tbody>
</table>

The ability of users to inspect the open source code produces “intelligent users”. Since users can review, discuss, debate, modify and enhance the source code, open source model users tend to understand these models at a much more theoretical and deeper level than is possible with proprietary models. The inability to inspect proprietary model source code by contrast tends to produce “button pushers” - users who know how to make the software perform an operation, but who have little or no concept of how the software performs that operation.

The lack of user understanding about the technical details of model program code inherent in proprietary model use has very damaging effects on the water resources profession. First, there is little basis for users of proprietary models to exchange ideas related to the theory or formulation of the model, since only the developers have access to the source code. Second, proprietary interests can impede idea sharing, since one model developer may use new ideas to advance their commercial interests at the expense of their competitors, rather than sharing the ideas. Third, proprietary models have little value in an academic environment, since model formulation and coding cannot be inspected and used for teaching.

Open source models by their very nature foster the exchange of ideas, since the source code can be reviewed and discussed. This characteristic is particularly valuable in academic settings, where the open source code can be
used as a teaching tool. And the absence of commercial interests eliminates the
incentive to maintain new ideas in secrecy for competitive advantage.

Another result of the commercial nature of proprietary software is
obviously higher cost to users. Since development, maintenance and support
are performed as commercial activities, those costs must be recovered through
the license cost and maintenance fees. From the developers' perspective, those
revenues must be sufficient to sustain the commercial enterprise. In many
cases the specialized nature of water resources software comprises a very small
market, in which it is difficult to sustain competitive commercial enterprises.

Commercial software developers are encouraged to continuously improve
the ease-of-use aspect of their software, as improvements in usability make the
software available to less technically sophisticated users and therefore can
translate directly into an increased user base (i.e. greater revenue). This is
another reason why commercial interfaces have become popular, and since
ease-of-use is primarily an attribute of the interface, this benefit applies equally
to both commercial models and interfaces.

19.5 Conclusions

Advantages and disadvantages exist for both approaches to model ownership
(open source versus proprietary commercial enterprise). The open source
model advantages generally outweigh the disadvantages relative to proprietary
models. Many of these advantages stem from the free access to source code
allowed by these models. Despite the modest increase in the use of commercial
models, it is fortunate that the open source models generally continue to enjoy
a dominant position in North American practice. However, open source models
will continue to advance and remain the preferred approach only if the water
resources engineering community continues to advocate their use, participate
in their development and enhancement, and maintain an active user base.

Proprietary software does not support user inspection and verification of
source code - open source software does. The advantages of open source code
are primarily realized in the computational engine, not in the interface, since the
user needs to understand how the model reads the input and performs its
computations, but not how the GUI software displays the input or results. In
contrast, many of the advantages of commercial software are realized primarily
in the interface, not in the computational engine. For these reasons, the best
software ownership scenario for most users of water resources models is one
where a selection of competitive proprietary interface products are available.
which employ a public domain computational engine. (However, a useful public domain GUI is valuable for occasional model users and academic teaching purposes.)

A final note to educators is warranted on the advantages of open source code models. The open source code only has value if engineers learn programming languages and coding techniques (not just spreadsheets). The authors encourage engineering educators to include these skills in their water resources engineering curricula, not to prepare their students to become professional programmers, but to ensure that they are knowledgeable users of water resource models.

References