Chapter 5

PRACTICAL INFORMATION AND EVALUATION CRITERIA FOR IRRIGATION PROGRAMS

M. Jurriëns (ILRI) & P.O. Malaterre (CEMAGREF)

5.1 The need for practical evaluation criteria

As we have seen in Chapters 3 and 4, there are many programs, of many types, for many purposes, varying in quality. Several classification systems or categories of available irrigation programs, mainly according to their subject or theme, were mentioned in the previous chapters (IRRISOFT, LOGID, ILRI). Making useful groups is not so easy, but evaluating and comparing them systematically (say, per group) in terms of properties and qualities, is even more difficult.

Computer programs have many facets, which actually have been addressed and should have been documented during the process of model building (or software engineering). During the various stages of model building (from conceptualisation to validation), many questions were answered and decisions made by the developers. However, assumptions, limitations, or specific solution techniques are not only of interest to the program developer, but may also affect the usefulness of the program for a practising irrigation engineer. This information tends to disappear in the marketing stage of the program, especially the limitations. Of course, one can buy or drive a car without being a car manufacturer or a mechanic, but essential information to make a choice between various models and types of cars must be available to the potential buyer/user. Such essential information is not standard available with irrigation programs, which makes it very difficult to compare and evaluate them, and make the right choice.

It would, therefore, be useful to develop a framework for evaluating programs, e.g. in the form of a systematic checklist of criteria. In the future, such criteria might be used by an international body to give irrigation software a rating and encourage the wider use of vetted programs. Compare, e.g., the International Groundwater Modelling Center, which started making inventories and developing criteria, and which now acts as an "evaluation, testing and clearing" house.

In this context, we may refer to Rogers et al. (1991), who presented evaluation and comparison criteria, especially for canal hydraulic models, at the ASCE Hawaii Conference. They stated that "model evaluation is intended to describe each program's capabilities, application, and usefulness." The established criteria were applied to six models. The task committee was originally set up to examine existing computer programs for their suitability and to foster communication among developers and users (Clemmens et al., 1991).

Another illustration of the need for irrigation software criteria is found in the Opening

session of a recent FAO expert consultation (FAO, 1994), the objective of which was said to be "to establish criteria to guide model development for the improvement of irrigation water delivery (...), taking into account the considerable development of irrigation and drainage software which has taken place over the last few years".

Therefore, we conclude that there is scope for further developing such criteria for irrigation software. We do this in the following Sections by first looking at existing forms of general program information. We then proceed to look at what has been done in terms of validation and evaluation. After that we propose a general information and evaluation format for irrigation software.

5.2 Existing work on general program information

- *The FAO expert meeting*

The aforementioned Expert consultation in Rome in October 1993 (FAO, 1994) suggested to address issues like:
- how flexible is the software?;
- how easy to learn?;
- how secure against the inexperienced user?

Other questions raised were: "Is there a need for development of new types of models?" and "How should dissemination be managed?". Furthermore, sustainability aspects were emphasized, addressing aspects such as staff training, staff motivation, software support and maintenance, improved communications and keeping farmers informed. Aspects of cost and organizational management required to collect the necessary field data were also mentioned. Although conclusions are not well-outlined in the proceedings, the forms that were used for describing the computer software presented at the consultation (contained in their Annex III) are useful. An example is shown in Table 5.1. The information is of a general type and is comparable to the brief ILRI inventory pages (Lenselink & Jurriëns, 1993). The pages are notably shorter than e.g. the Software Description Pages of IRRISOFT (see Chapter 2). Mainly containing general information, they definitely assist in making a first classification or selection, but they do not contain sufficient information for a thorough evaluation and comparison.

- *IRRISOFT*

The most recent attempt at presenting a format for describing irrigation software is found in Chapter 2, where an IRRISOFT structure for the Software Descriptive Pages is mentioned (see Table 2.1). There is no need to repeat that structure here, but it shows a rather comprehensive approach at describing the most important general software information, so that potential users can make a first selection from the many existing programs. In a further development stage of IRRISOFT, the format may change to include more details, e.g. also on assessment, comparison and evaluation of properties and qualities.

Table 5.1 Example of a 1993 FAO software descriptive form

Software name:	SIC
Software type:	Hydraulic simulation
Functions:	Simulates system behaviour to identify apppropriate operational strategies. Evaluates effects of changes to system parameters
Suited to:	Steady and unsteady flow conditions Branched and networked systems
Brief description:	Three principal programs carry out topographic generation, steady flow computation and unsteady flow calculations respectively. Model calibration assisted by a module which calculates discharge coefficients and roughnesses from field data. Steady flow calculation can be performed on an type of network. Unsteady flow conditions at present only possible on non-looped networks. Versions in English, French, Spanish
Use:	Menu-driven
Input:	User-friendly interface for: topographical data and network definition; seepage rates, flows, gate openings Structure regulation rules need to be written in special modules (in FORTRAN) which can be linked with the program.
Output:	Graphical or numerical interfaces or results files. Water levels, flow velocities, discharges at points throughout system. Comparisons between actual and predicted situations.
On screen help:	Yes
Language:	FORTRAN, TURBO PASCAL
Graphics:	Yes
Other reqs: (software)	No
Hardware:	IBM-PC/PS2 or compatible. Minimum 1 Mb RAM, 20 Mb HD. Maths coprocessor
Ref paper:	P Kosuth. Application of a Simulation Model (SIC) to Improve Irrigation Canals Operation: examples in Pakistan and Mexico
Contact:	P. Kosuth, Head, Irrigation Division, CEMAGREF, 361, rue J.F.Breton (BP 5095), 34033 Montpelier Cedex 1, France
Tel:	
Fax:	33-67635795

- *The previous ILRI inventory*

The ILRI inventory of 1993 mentions some practical usability criteria, particularly concentrating on four aspects, i.e. hardware requirements, user-friendliness, the manual, and availability. Hardware requirements do not often pose a real problem these days, although some programs may require extra memory, a digitizing tablet, a flat-bed scanner

or a special plotter. Under user-friendliness the following points were raised:
- program should be self-explanatory on screen;
- screen lay-out should be logical and clear;
- basic actions must be under commonly-used keys;
- program should be fool-proof;
- interactive data input is preferred above batch processing;
- file handling should be straightforward;
- graphical output should be included whenever possible.

Despite the requirement that a program should be self-explanatory and contain on-screen help, a good manual should accompany the program. Such a manual should at least include a clear introduction, background theory, the program structure, a user instruction, an example case, a common error listing, and a clear index.

The availability of a program was discussed in terms of being adequately advertised, being quickly sent when ordered, being reasonably priced, and having a fixed contact point.

5.2 Irrigation software validation

- *Software development*

As already mentioned in Section 5.1, there is a definite link between software users and software developers on the topic of software quality. It may, therefore, be useful to take a brief look at some relevant issues that are mentioned in a few software engineering literature. Deutsch and Willis (1988) mentioned fifteen required software qualities, mainly from the developer's point of view, but also with interesting points for the end user (compare Jurriëns & Lenselink, 1992). In the initial stages of program building, questions like: Who is to use the program?, What is the basic objective?, What input data are required?, Which results can be expected?, Why is the computer model necessary? clearly are questions that have a bearing on the purpose, properties, and qualities of the final product. Moreover, after the computer programming or implementation has taken place, one may expect the program to:
- use efficient code;
- have adequate error traps;
- give technically correct results;
- possess robustness;
- have been extensively tested;
- require reasonable input;
- return default values where possible;
- return useful output of reasonable detail and format.

In addition, the program should be able to run on different computers (portability), and should be written and documented in such a way that both a programmer and a user find it "friendly". Programmer friendliness has to do with maintenance and flexibility, and includes internal and external documentation, logical and modular lay-out, use descriptive

variable names, have built-in debugging aids, etc. User friendliness has more to do with the ease with which a normal computer user can apply the program. More than a decade ago, Ingels (1985) already mentioned that a program must:

- be interactive;
- be menu-driven;
- have a reasonable response time;
- have a reasonable amount of input and output;
- let the user always know what to do next (+ help);
- have an adequate user manual;
- have a reasonable price;
- not require excessive learning time.

Such requirement lists, especially the earlier ones, are not always adequately structured, but they assist in forming an opinion of aspects to include in a more comprehensive framework for evaluation of irrigation software. We shall consider such a framework in the Section 5.5.

- *Software validation*

In fact, if we are trying to find software criteria, we are busy with the last stages of software development, for which we can distinguish the following seven stages: conceptualisation, (mathematical) model building, programming, verification, calibration, validation, and evaluation. We have already referred to some of these stages above. Our quest for criteria covers the validation and evaluation stages. In some instances, these two stages are lumped together under "validation". In that case, program validation is understood to be the process of testing and documenting the quality of a computer program, in relation to its intended applications and the physical system it represents. Others make a distinction between validation, i.e. testing the program results against some independently measured data, and a subsequent evaluation, in which one wants to assess a program's applicability and usefulness. This evaluation is exactly what we want to achieve, and for which we are trying to find a suitable, structured format.

In other sectors of industry it is not uncommon to require a validation document as part of an industrial product, which could also be applied to the software industry (for major packages). Some hydraulic institutions, who produce software in-house, have been considering such a validation document (which may ultimately lead to certification). Standardization of such a document has been advocated, and a possible format has been laid down. It may be useful to illustrate our discussion with the contents of a validation document as produced by Hydraulics Research Wallingford for their MIDAS program version 1.1 in March 1992. The contents page of this document is reproduced below in Table 5.2.

It can be seen that some general information about the model is given first, after which the aims of the validation are described in detail, before the validation in test cases is reported. However, this standardized approach to a validation document shows that our search for an evaluation framework is not a loose idea, but that it has roots in the more

Table 5.2 Sample contents page of a validation document

general process of software development. Merging ideas from other from such areas with our own can lead to a better evaluation system for irrigation programs.

5.4 Existing work on evaluation criteria

- *ASCE assessment and evaluation*

Let us take a closer look at the ASCE task committee's criteria, mentioned above. For their canal simulation programs, Rogers et al. (1991) distinguished between three main types of criteria:

- those dealing with the technical merits;
- those related to the modelling capabilities;
- those qualifying user considerations.

For each of them, further details were discussed as shown in Table 5.3.

Table 5.3 ASCE canal model evaluation and comparison criteria

- TECHNICAL MERIT
  - computational accuracy
  - numerical solution criteria
  - robustness
  - initial conditions
  - internal + external boundary conditions
  - special hydraulic conditions
- MODELLING CAPABILITIES
  - system configuration
  - frictional resistance
  - boundary condition types
  - turnouts
  - operations duplication
  - automatic control
  - miscellaneous limitations
- USER CONSIDERATIONS
  - user interface
  - documentation and support
  - direct costs
  - indirect costs

Although these criteria were specifically meant for their canal simulation programs, the approach is useful. It has yielded the inspiration for the extended and modified framework presented in Section 5.5.

- *CEMAGREF*

Also at CEMAGREF, the French ICID committee is working on further detailing, testing, validating, evaluating, and comparing irrigation software. Mainly as a consequence of the earlier comparisons of the ASCE task committee mentioned before, one is concentrating on non-steady canal flow models first. For the SIC (Simulation of Irrigation Canals) model, a list of aspects that may help to qualify the model has been prepared, and one intends to compare other French programs (like Elicsir) with them. A format that is currently in use has been translated and is reproduced in Table 5.4. This format can be used to describe the various programs in a number of sections, like: Data input, Calculations, Output, Documentation, Special features, and Hardware requirements.

Table 5.4 A CEMAGREF format for comparing non-steady canal flow models

INPUT	SIC	Elicsir	
Interactive data input	Yes		
Input from ASCII data files	Yes		
Automatic data checking	Yes		
Real geometry (point-by-point canal reaches)	Yes		
Easy modelling of trapezoidal canals	Yes		
Automatic classification of branches	Yes		
Automatic interpolation in reaches	Yes		
Names for branches	Yes		
Names for nodes	Yes		
Names for reaches	Yes		
Library of cross-regulators	Yes		
User-defined cross-regulators	Yes		
Parallel cross-regulators	Yes		
Cross-regulator manipulation	Yes		
Library of offtake structures	Yes		
User-defined offtake structures	Yes		
Parallel offtakes	Yes		
Offtake regulation	Yes		
Initial steady-state water level	Yes		
Initial transient water level	Yes		
Maximum number or branches	80		
Maximum number of reaches	600		
Maximum number of cross-regulators	200		
Maximum number of offtakes	80		
CALCULATIONS			
Steady-state	Yes		
Non-steady (transient)	Yes		
Complete Saint Venant equations	Yes		
Solution scheme	Preissmann		
Implicit solution	Yes		
Solution technique	Double sweep		

Variable time step during calculation	No		
Variable distance step	Yes		
Discharge and water elevation in all reaches	Yes		
Flooding	No		
Location of drop	No		
Dry water front	No		
Movable drop	No		
Pressurized flow	Yes		
Mesh (joints and bifurcations)	Yes		
Discharge at offtakes	Yes		
OUTPUT			
Screen: tables	Yes		
Screen: graphs	Yes		
Printer: tables	Yes		
Printer: graphs	Yes		
File: ASCII	Yes		
File: graphs	Yes		
Links with other programs: dBase	No		
Links with other programs: SIG	No		
Links with other software: CAO	Yes		
Water-distribution performance indicators	Yes		
Output for all points and times	Yes		
Water-level movement display	Yes		
DOCUMENTATION			
User manual	Yes		
Background theory manual	Yes		
Various languages	Yes		
Help screens	Yes		
Test cases	Yes		
SPECIAL FEATURES			
Library of gate settings	Yes		
User-defined gate settings	Yes		
Sediment transport	No		
Pollutant transport	No		
Salt transport	No		
HARDWARE REQUIREMENTS			
Computer type	PC		
RAM	640 kB		
Hard disk space	5 MB		
Operating system	DOS, Windows		

5.5 Proposal for a modified evaluation framework

Taking into account the results of the FAO and ASCE meetings, software engineering considerations, and other work done, as described in Sections 5.2-5.4, we propose a

modified evaluation framework. Its broad set-up is largely the same as that of Rogers et al. (1991), which we have modified to agree with earlier remarks on the subject (see also Chapter 1).

We start with a category 'General information', containing the name of the program, the contact address, relevant literature, etc., combining items from the FAO software descriptive forms (Table 5.1) and the IRRISOFT software descriptive pages (Table 2.1), but excluding items that are falling in the other two categories, i.e. Properties and Qualities.

The major distinction is between 'Properties' and 'Qualities'. Properties then relate to the more factual information ("What can a program do?"). Under Properties we have added 'Scope and purpose' and the mechanical (hardware) requirements. 'Purpose and scope' includes some aspects of Rogers' modelling capabilities.

The second group (the 'Qualities', i.e. "How does a program do it?") concern the 'Program qualities' and the user-friendliness. A distinction has been made in Program qualities between 'Theoretical quality', which refers to the conceptual and model-building phases in software engineering (theories, assumptions, mathematical representation), and the 'Technical quality', which concerns the implementation or the programming of the model. 'User qualities' are a very important category, containing aspects which immediately concern the end user of the program. This framework is shown in Table 5.5. Details of this evaluation framework are discussed below.

Table 5.5 Proposed irrigation software evaluation framework

General information		- program name - made by - cost - reference person - programming language - manual availability - key refernce publication
Properties	*Scope and purpose*	- subject - purpose - capabilities/options - limitations
	Hardware requirements	
Qualities	*Program qualities*	- theoretical quality - technical quality
	User qualities	- interface - documentation - availability

- Properties: Scope and purpose

As a first item under 'Scope and purpose' one can find for which *subject* the program can be used, and *what it can do* for that subject. The indication of the subject should be sufficiently detailed and clear. "Surface irrigation" is not adequate for a specific furrow irrigation design program; the mentioning of "furrow irrigation" and "design" are essential. It should further be mentioned if it includes cut-back, blocked-end or re-use options and if the program concerns one furrow or the complete field lay-out. A brief indication of required input and expected output often makes the subject clearer.

A second item to be mentioned is the *purpose* for which the program has been made. A program can be meant explicitly for planning, design, operation, evaluation, or training. Some programs are simple calculation tools, others are simulating a process, to be used for any purpose. Of course, a design program can be an instructive training tool, but specifying the *target group* for which the program was developed assists end users in making a choice.

Under 'Options/capabilities' information can be find about *input/ouput* options of the program, if the *units* can be changed, if subject-specific *modes* can be chosen, etc. In practical terms, this is a further detailing of the subject, combined with computer-specific items. Special distinguishing *features* can be mentioned here ("... produces daily, weekly and monthly totals in tabular and graphical form..").

A clear statement on the *limitations* of the program should be included, if it were only to avoid disappointed buyers/users. Such limitations can have to do with the subject, the purpose, and the options mentioned above. They can also indicate limits of data ranges or scale, or can state underlying assumptions and boundary conditions. Examples are: "... this program is not suitable for design purposes, but returns order-of-magnitude estimates only ..."; "...the program only considers uniform soil condition...";"...the program accepts monhtly average values only...").

- Properties: Hardware requirements

Under this heading, the *operating system* must be specified (MS-DOS 6.0 and higher, Windows 95). Also the necessary and recommended *processor* (Pentium 100 MHz), the required free *memory* for installing and running the program should be stated. One also would like to know if a *hard disk* is required to unpack/install the program, whether a (special) *printer* is needed, and if a certain *graphics or sound card* is necessary. Whether or not a particular *keyboard*, *monitor*, or *mouse* is required is also useful to know. Further possible items to include are mentioned in Chapter 3 (Figure 3.4).

- Qualities: Program qualities

Under 'Theoretical quality' we expect information on the underlying *theory* of the program ("... based on a full solution of the St. Venant equations..."). Virtually all irrigation programs are based on a mathematical modelling of a part of reality, and it is important

to know whether the *model approach* uses simple regression equations or more universally applicable physical laws.

Apart from this conceptualisation, one would also like to assess the chosen *modelling approach*, i.e. the mathematical approach used. Stating which (type of) algorithms and physical or statistical laws were applied is useful, so that the user can judge their acceptability.

In the implementation phase of modelling, bugs could have entered and therefore it is important to know if the program has been *de-bugged* and *verified* to give correct results for test cases. Test case results are mainly a software engineer's worry, but a user would like to know about the most recent tests.

Under the heading 'Technical quality' we mainly expect information on the chosen numerical solution technique, which affects a number of criteria, such as *correctness*, *accuracy*, *stability*, and *convergence*. Correctness is self-explanatory. Accuracy and stability deal with unavoidable rounding or truncating error in the many calculations, especially if differentiation or integration have to be done numerically. Smaller (time) steps lead to a greater accuracy, but the cumulated error may become so large that it approaches the solution, in which case the stability is lost. Convergence is another requirement is numerical iterations: we would like to know if the program will always give a solution (implicit solution schemes will, explicit ones may not).

A further technical quality relates to *input sensitivity* (are input ranges limited, or does the program also give a solution for freak values), which quality is also referred to as *robustness*.

A last technical quality that a user would like to know about is whether any *calibration* and/or *validation* has been done, and what the results thereof were. Calibration refers to the testing of the program versus measured data, after which adjustments may have been made. In the validation stage such adjustments are not made. Compare the remarks on validation made in Section 5.2.

- Qualities: User qualities

The user qualities are often neglected, but they form the link between the program and the user and as such are very important for its application (also see Chapter 6). One could also describe these user qualities as the degree of user-friendliness. We distinguish three groups of aspects, i.e. the *user interface* (on the computer), the *manual*, and the *availability* of the program.

For the **user interface**, we can specify the following aspects: accessibility, clarity, program handling, file handling, input and output. For each of these aspects, we have listed a number of requirements below:

* *Accessibility*
  - easy install, start, stop;

* *Apparent simplicity*
  - clear program structure;
  - clear and consistent menus;
  - easy to to browse, get back, get out;
  - set of default data (standard file);
  - clear input/output;

* *Program handling*
  - screen help (meaning/purpose);
  - common key operations + instructions;
  - clear terminology;
  - clear screens;
  - error messages;
  - time to learn/manual/training;

* *File handling*
  - retrieve and save;
  - dos/windows options;
  - import, export, convert;
  - track record;

* *Input*
  - consistent option selection - input;
  - interactive/on screen;
  - clear meaning/purpose;
  - message on ranges;

* *Output*
  - clear screen;
  - primary and secondary;
  - report, tables, graphs;
  - save/print/plot.

The **documentation** quality mainly concerns the user manual (in contrast to the programmer's manual). Although the necessity of a clear manual has often been stressed, a number of programs still do not have them. A good manual should "document the objectives, target groups, relevant current developments, the methodology and the process of program development, the background theory, the use of approximations and constants. It should also explain the use of the program step-by-step and point out any less common uses. At least one worked example should be included, the data of which should already be available on the distribution disk", as stated in the ILRI inventory of 1993. A good manual should e.g. contain an introduction, a chapter on the background theory, a

summary and explanation of the program structure, a section on how to run the program (operation), one or more worked examples, data ranges and a good index.

Another user concern is the **availability** of the program. Many of the irrigation software packages are non-commercial, and therefore are not officially *marketed*. The existence of a certain program often follows from a journal article, from workshop proceedings, or from correspondence between a selected group of people. Making an inventory and adding qualities should help to overcome this problem to a certain extent. IRRISOFT (Chapter 2), LOGID (Chapter 3) and the ILRI inventory (Chapter 4) may certainly help.

Another availability aspect is the *price* of a package. Development costs of the larger packages are high (labour-intensive), and commercial institutions (consultancy firms, publishers) by nature want to sell their products at a profit. Many publicly-funded research and educational institutions do not have this urge and make programs available at nominal cost only (although privatization unfortunately leads to reversing this trend). Apart from the purchase price, there are also indirect costs which need to be invested in learning time, data collection, etc.

Under the availability heading one can also think of the **support** that is available for a software package; a name and an (e-mail) address where further information can be obtained, where queries are answered, and where updates are made (and made known).

References

Clemmens, A.J., W.R. Walker & R.S. Gooch, 1991. Irrigation canal system unsteady flow modelling. In: W.F. Ritter (ed.): Irrigation and drainage, Proceedings of the 1991 National conference, Honolulu, Hawaii, July 22-26, 1991: p. 231-237

Deutsch, M.S. & R.R. Willis, 1988. Software quality engineering - a total technical and management approach. Prentice Hall, Englewood Cliffs

FAO, 1994. Summary report, conclusions and recommendations. In: Irrigation water delivery models. Proceedings of the FAO Expert consultation, Rome, 4-7 October, 1993. Water report #2: p. 1-10

Ingels, D.M., 1985. What every engineer should know about computer modelling and simulation. Marcel Dekker, New York

Jurriëns, M. & K.J. Lenselink, 1992. User-oriented irrigation software for micro-computers. In: Annual report 1992, ILRI, Wageningen, p. 41-51

Lenselink, K.J. & M. Jurriëns, 1993. An inventory of irrigation software for microcomputers. ILRI Special report, Wageningen. 172 p.

Rogers, D.C., W. Schuurmans & J.W. Keith, 1991. Canal model evaluation and comparison criteria. In: W.F. Ritter (ed.): Irrigation and drainage, Proceedings of the 1991 National conference, Honolulu, Hawaii, July 22-26, 1991: p. 323-329