The object-oriented paradigm (OOP) provides a methodology for resolving a crop simulation system into subsystems and processes so that a modeler can design objects to simulate their behaviors. Many objects are also devoted to producing object-oriented user interfaces to simplify operation of the simulator. How to integrate these objects on a higher level and how to make models and user interfaces communicate efficiently are the issues addressed in the current research. The proposed model-document-view (MDV) architecture provides the modeler with an extension of OOP programming in crop modeling and a flexible scheme of system management and efficient data communication between simulator and user interface. The main advantage of the MDV architecture is that it separates the domain model, data management, and user interface but allows them to communicate efficiently. By using this architecture, the whole crop modeling system can also be integrated into several MDV groups. The system structure is clear and easy to administer, and communication between objects is more efficient. Here we report our experience with MDV architecture in constructing a rice (*Oryza sativa L.*) model called Visual Rice Growth Model (VRGM). The objects in VRGM were clustered into three MDV groups. The MDV paradigm was shown to meet the requirements of massive data management and displaying in a crop modeling system. The architecture can be used in any modeling system based on OOP.

Crop models are concurrently evolving in two divergent directions: easier interfaces for users and more intricate and integrated processes in the model itself. More and more subsystems, physiological processes, and variables have been incorporated into crop models, which are becoming increasingly complicated. Despite the use of structured programming approaches, conventional whole-system models are typically large, complex, and expensive to create, maintain, and extend. Since the early 1990s, the computer software industry has experienced what some observers consider a cultural revolution in the form of the object-oriented paradigm. Object-oriented programming offers a solution to the single use and expense of whole-system models. The object-oriented paradigm helps manage model complexity and reduces expense. Of course the complexity of models reflects the real complexity of plant–soil–atmosphere–management interactions, and so some complexity is unavoidable.

Developing a user interface that incorporates tools to simplify operation of the crop simulator has also been recognized as an essential step toward increasing the utility of crop simulation software and decision support systems (Landivar et al., 1989; Cox, 1996). Command-line interfaces have been replaced during the last decade with graphical user interfaces (GUIs) based on graphics (windows, icons, menus, and pointers) instead of text. Several crop model GUIs are described in the literature (e.g., Hoogenboom et al., 1994; Van Evert and Campbell, 1994; Waldman and Rickman, 1996; Testezlaf et al., 1998; Acock et al., 1999). Some are built specifically for a single crop model, some for a family of models, and some generally for any crop simulator. User interface design and implementation is a growing field in software engineering. The software industry is currently migrating from GUI to object-oriented user interfaces (OOUl) that provide direct manipulation of objects instead of functions to perform. Many objects have to be designed to produce user interfaces for displaying simulation results, entering weather data and agronomic data, manipulating options, and so on. So a crop simulation system contains many objects for models and user interfaces.

Object-oriented programming defines no concepts to organize objects produced by itself, only division of duty of these objects. But Visual C++ provides such a programming paradigm, called document-view architecture as an organizing principle, to separate data display and data management (Zarattian, 1998). In crop modeling software, the paradigm can be extended to MDV architecture to separate domain model, data management, and user interface. We believe that the MDV approach in the design and operation of modeling software can help because it (i) improves the robustness, extendibility, and reusability of software systems by separating the model, data management, and user interface; (ii) allows the addition of new objects without the need to change system structure; (iii) makes communication...
between objects more efficient, (iv) increases modeler productivity; and (v) reduces redundant code.

The overall objective of our work is to define and implement a modeling framework of OOP, MDV architecture, that can be used in all OOP-based modeling software. Use of this framework should make the crop simulator and user interface communicate efficiently and combine the multiple objects into an efficient system. We report here on our experience with the MDV paradigm in constructing a simulation model, VRGM.

MODEL-DOCUMENT-VIEW ARCHITECTURE

The goals of the MDV paradigm are to (i) separate the crop model, data management, and user interface; (ii) make the data of model and user interface communicate efficiently, with the document as an intermediary; (iii) cluster the objects for simulating, managing, and displaying the same data into a distinct group while allowing data communication between these objects; and (iv) reduce unnecessary complexity in modeling system.

Our perception of the advantage of this paradigm is based on our experience of engineering VRGM with Microsoft Foundation Classes (MFC) in Visual C++ 6.0 (Microsoft Corp., 1998). Applications of MFC use the document-view architecture to manage data, file formats, and the visual representation of data for users. The document-view architecture separates data from view, simplifies the application, and reduces redundant codes. In this architecture, a document object reads data and writes them to persistent storage. A separate view object manages data display, from rendering the data in a window to user selection and editing of data. The view obtains displayed data from the document and communicates any changes back to the document. The developer can easily add application-specific code to this architecture. The key advantage of using the document-view architecture is that it easily supports multiple views of the same document (Kruglinsk et al., 1998). Gauthier et al. (1999) separated the domain model and the user interface to keep them independent in designing their generic and object-oriented framework for crop simulation. Several data communication methods between simulator and user interface are described in the literature (Van Evert and Campbell, 1994; Testezlaf et al., 1996; Acock et al., 1999). In crop modeling, the document-view paradigm can be extended to a MDV paradigm to meet the requirements of bulk data management and display within the context of an integrated modeling system.

The Model-Document-View Paradigm

In MDV architecture, the view is the user interface used to display input or output data. The document acts as a transmission agency of data exchange between model and user interfaces. The MDV architecture consists of one or more models, one document, and one or more views (Fig. 1). Model document view is a flexible concept in which a modeler may set fewer models and one document to administer it easily, and the number of views depends on the amount and types of data in the models.

The document is used to load, store, and control output and input data in models. A document can be associated with several views to render images of different parts of the data in the document. The document offers interfaces consisting of document variables and simulator options. The end results of numerical simulations are stored in the document for display through views. Intermediate results and simulation variables are stored in the model itself to simplify the design of the document. At the same time, user operations are interpreted as data input or manipulation of simulation options, and changes in the document are fed back to views. Weather, soil, agronomic, and other data are stored in the document for utilization by the simulation model(s). Every type of data can be saved, opened, previewed, and printed, corresponding with the appropriate command.

A view is attached to a document and acts as an intermediary between the document and the user: The view renders an image of part of data in the document on screen or printer or interprets user input into operations on the document. Because there are massive and multiple types of data in modeling system, many views are designed to meet the needs of data display or input. The number of views depends on the type and the amount of data in the model and the design of a view. For instance, one or several dialog boxes are enough for the model dealing with agronomic data while several spreadsheets have to be included into MDV for the model that is related to weather data. Control of which view is visible and on top of the others is managed at the operating system level.

Fig. 1. Relationships between objects in the model-document-view architecture. Arrows indicate data communication between objects. Communication between model(s) and document is bidirectional while data flow between document and views may be unidirectional or bidirectional.
The working procedure of a MDV group can be described as follows: A user enters the model’s initial data through the view and invokes the model; these data are then transmitted to the document. The document then instantiates the model object and manipulates the simulation process. The document stores the end results and at the same time feeds the simulation results back to view(s) attached to it. The view then translates the results of the model into a display.

The MDV paradigm can be readily implemented in OOP-based programming systems such as Borland C++, Visual C++, Delphi, or GCC. The view contains functions to manipulate the document and part of data in it while the document contains functions to handle the models. So the view can manipulate simulator through the document while the MDV group can efficiently transmit data from view to model and from model to view. So the user manipulates the model indirectly: Changes in the view alter the document, which in turn manipulates the model. Meanwhile, results from the model flow efficiently to the document and from there to display in one or more views. Derived from MFC, the MDV paradigm is easily realized in Visual C++ using MFC, in which the modeler need only establish interfaces within the document to manipulate models. The links between the document and the view are automatically installed by MFC. In other programming environments, MDV can be implemented as described above.

**Integration of the System with Model-Document-View Architecture**

The whole modeling system can be integrated by the MDV paradigm (Fig. 2). The massive objects in a modeling system can be divided into several groups by the architecture. A MDV group includes one or several models, a document, and several views. The objects in a modeling system can be assembled into several MDV groups, minimizing the complexity of the modeling system. The document transmits data between the model and the view within a MDV group. In the whole system, the documents in different groups share their data with each other, so different groups can share their data through their documents. Different models can acquire data from other MDV groups easily. In the modeling system, a MDV switcher has to be installed to coordinate the MDV groups to shift operation from one MDV group to another in accordance with the user’s commands. The MDV switcher is a function or an object at the level of MAIN program to organize the whole modeling system by handling views in system. The data communication method in MDV paradigm decreases the unnecessary interfaces between objects, reducing redundant code. Built by MDV architecture, the modeling system structure is clear and easy to administer, and communication between objects is more efficient. When a new model is added to the modeling system, a modeler can easily add a group to the system, extending the whole system without having to change system structure. This is useful for construction and maintenance of a complex agricultural modeling system. The MDV architecture can be applied to any modeling system based on OOP. The results similar to OOP can be obtained through a carefully designed procedural analysis of an agricultural system (Van Evert and Campbell, 1994), and results similar to MDV can also be attained by careful organization of components and subroutines in structured programming. However, MDV is based on OOP and has larger structure than OOP, so attaining MDV by procedural programming will be more difficult, especially in large modeling system.

**PROGRAM DESCRIPTION**

**Developing the Visual Rice Growth Model**

Much of the design of VRGM is based on the Rice Status Simulation Model (RSSM), developed to simulate rice growth and yield under the conditions of ample water and nutrients (Zou et al., 1991a, 1991b). Rice Status Simulation Model itself is partly based on work done on Rice Clock Model (Gao et al., 1987, 1992). Rice Status Simulation Model considers that water stress is negligible because the supply of irrigation water in the Yangtze River Valley is sufficient to meet the rice’s needs. Rice Status Simulation Model consists of six submodels dealing with (i) rice phenology, (ii) leaf age, (iii) tiller density, (iv) leaf area index (LAI), (v) daily gross photosynthesis, and (vi) biomass accumulation. Because most of Chinese weather stations have no records of solar radiation, RSSM also involves a submodel to calculate daylength and to derive solar radiation from latitude, daylength, and sunshine duration (Zou et al., 1993a). Compared with CERES-Rice, ORYZA-1, and other rice modeling systems, its data requirements are modest (Table 1), and it assumes optimum management

![Fig. 2. Diagram of a modeling system integrated with the model document-view paradigm. The modeling system is divided into several groups. Arrows represent data exchange between objects. Crop models represent one or several classes or objects in the system. The document acts as intermediary in data communication within a group. The documents in different groups share data with each other.](image-url)

**Table 1. Selected data requirements for the Rice Status Simulation Model (RSSM).**

<table>
<thead>
<tr>
<th>Input data</th>
<th>Agronomic data</th>
<th>Station information</th>
</tr>
</thead>
<tbody>
<tr>
<td>Weather data:</td>
<td>Agronomic data:</td>
<td>Station information:</td>
</tr>
<tr>
<td>Daily mean temperature</td>
<td>Sowing and transplanting date</td>
<td>Latitude</td>
</tr>
<tr>
<td>Daily sunshine duration</td>
<td>Seedling planted per square meter</td>
<td></td>
</tr>
<tr>
<td>Daily solar radiation</td>
<td>Maximum leaf area index</td>
<td></td>
</tr>
<tr>
<td>Tiller density</td>
<td>Total leaf number</td>
<td></td>
</tr>
<tr>
<td>Leaf area index</td>
<td>Variety type</td>
<td></td>
</tr>
<tr>
<td>Gross photosynthesis</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Biomass accumulation</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
practice. It simulates and predicts the main characteristics of rice growth day by day, which is helpful for making management decisions. But water balance and N supply and uptake are not taken into consideration in RSSM. The model parameters were determined with data collected from 12 counties in Hunan Province during 1984–1991. Varieties were grouped into eight variety types: medium- and long-duration early rice; short-, medium-, and long-duration medium rice; and short-, medium-, and long-duration late rice. It was shown (Zou et al., 1991a, 1991b) that all varieties within a variety type have the same parameter values in the RSSM phy-

Fig. 3. (A) Spreadsheet in weather view and (B) broken line graph of simulation result.

nological model. Rice Status Simulation Model provides two submodels for extracting coefficients for the tillering and leaf age submodels from at least one year’s worth of local data for the desired cultivar, making it easy for the user. Rice Status Simulation Model was validated using data from Hunan Province and two neighboring Provinces (Jiangxi and Hubei) also in the Yangtze River Valley (Zou, 1993a, 1993b).

In accordance with the principles of OOP, six subsystems in the rice production system were abstracted and encapsulated in six objects: CTime, CWeather, CPhenology, CPhotoSynthesis, CCoefficient, and CAgro-
nomicData. Within VRGM, time is represented as day of year. CTime keeps track of time and supplies other objects with information about the current time. It also transforms between day of year within the model and Gregorian day for users. CWeather receives data (such as mean temperature, sunshine duration, and solar radiation) and calculates daylength, accumulated temperature, and solar radiation. CPhenology mainly simulates temperature and daylength effects on rice development and phenological events, but leaf age, tiller density, and

Fig. 4. (A) Image simulation interface and (B) management decision provided by Visual Rice Growth Model.
LAI can also be estimated on the basis of effects of degree days on leaf expansion and tillering. CPhoto-Synthesis deals with rice daily photosynthesis, carbohydrate partitioning, and biomass accumulation. CCoefficient analyzes experimental data to extract the needed coefficients for the tillering and leaf age submodels. Finally, CAgronomicData handles agronomic data, such as maximum LAI and density of seedling transplants.

**User Interface and Views**

**User Interfaces for Data Input and Output**

The quantity of data used in simulation models is massive. For example, weather data includes daily mean temperature, incident radiation, and sunshine duration while rice growth data includes daily leaf age, LAI, and photosynthesis. In most simulation models, these data are saved in fixed files written by the program and then read later using commercial software. In VRGM, data are inputted and displayed in spreadsheet cells (Fig. 3A). Input data can be edited and browsed in the spreadsheet, and the output data are made read-only by changing the property of spreadsheet. Model outputs can also be shown with a broken line graph in the dialog box (Fig. 3B). Data with just a few points, such as sowing date, transplanting date, and variety type, are entered in a dialog box.

**Visual Simulation and User Interface to Images and Graphs**

Under conditions of ample water and nutrients, normal climate, and optimum management practice, rice photographs of individual and population at different stages were taken with Olympus digital camera (Olympus Optical Co., Ltd., Shinjuku-ku, Tokyo) at 3-d intervals, producing a large number of photograph files, which were saved in BMP format. Visual Rice Growth Model shows individual and population photographs at different stages corresponding to the simulation results (Fig. 4A) and suggests management decisions (Fig. 4B).

Hermite curves (Hearn and Baker, 1997) were employed to draw rice leaves and spikelets. The initial point and end point stand for the base and apex of leaves and spikes, respectively (Fig. 5A). Green is filled between two Hermite curves, representing a leaf. A single Hermite curve (standing for spike-stalk) overspread with an ellipse (standing for grain) represents a rice spike (Fig. 5B). The rice individual at different stages is drawn dynamically to show the results of numerical simulation. Users can compare simulation results with the actual state of rice growth in the field and make cultivation decisions easier.

**User Interface for the Whole System**

In the user interface framework of VRGM, two frames are installed: The left frame is a data directory, and the right frame is the data area, consisting of an active view and other hidden views. In the data directory, tree control is used to indicate the entire systematic structure of VRGM. A tree control is a window that displays a hierarchical list of items (such as directories). Each item consists of a label and an optional image, and each item can have a list of subitems associated with it. In the data directory, the environmental data, rice growth characteristics, extracted model coefficients, and image and graphics simulation are set as the nodes of the tree control, with every type of data being a leaf of nodes. Each node maps to a group of views in the data area. For example, the node of rice growth charac-
Fig. 6. The system structure of Visual Rice Growth Model (VRGM). The VRGM consists of three model-document-view (MDV) groups integrated by MDV architecture. The contents in frame window are visible to user. Data directory acts as a system handler.

Document and Data Management

Three documents—Weather, Rice, and Coefficient—were installed to manage data in VRGM. The Weather document manipulates CTime and CWeather and connects them with views to show or input weather data. Similarly, the Rice document administers model's initial data and the simulation results by controlling CPhenology, CPhotoSynthesis, and CAgronomicData and connects them with views for displaying simulation results as graphs, photographs, and graphics of rice growth characteristics. The Coefficient document connects CCoefficient and the views to input experimental data and to show extracted coefficients, and it also acts as the intermediary between Coefficient and these views. The three documents share their data directly with each other and with models and views. Additionally, the documents save, open, print, and preview data according to the user's command.

System Structure

The classes in VRGM were integrated with MDV architecture. The VRGM includes three MDV groups—Weather group, Rice group, and Coefficient group—and a system manipulator (Fig. 6). The Weather group coordinates classes of simulating, managing, and displaying weather information; the Rice group orchestrates classes of simulating, managing, and displaying rice growth characteristics; and the Coefficient group administers classes of entering experimental data and classes of extracting and showing model coefficients. The three MDV groups exchange data among each other via documents. The data directory acts as MDV switcher that shifts operation from one MDV group to another according to the user's command by activating one view and hiding the other views.

In VRGM, each type of data set can be stored in a data file. The interrelated data files are combined into a group that is called a project. Usually a project encompasses several data files that store daily weather data, daily rice growth characteristics data, or annual experimental data for extracting coefficients of tillering and
Table 2. Air temperature and sunshine duration during the 1999 and 2000 growing seasons.

<table>
<thead>
<tr>
<th></th>
<th>April</th>
<th>May</th>
<th>June</th>
<th>July</th>
</tr>
</thead>
<tbody>
<tr>
<td>Year</td>
<td>F† M‡ L§</td>
<td>F M L</td>
<td>F M L</td>
<td>F M</td>
</tr>
<tr>
<td></td>
<td>Mean air temperature</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td>17.2 17.1 18.4</td>
<td>20.6 22.4 21.6</td>
<td>24.8 24.9 24.0</td>
<td>25.7 25.0</td>
</tr>
<tr>
<td>2000</td>
<td>17.6 17.3 18.2</td>
<td>20.9 24.0 23.6</td>
<td>24.6 25.1 28.0</td>
<td>29.1 29.9</td>
</tr>
<tr>
<td></td>
<td>Mean sunshine duration</td>
<td>h d⁻¹</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td>5.12 3.33 2.59</td>
<td>5.27 5.5 3.85</td>
<td>3.1 6.22 0.96</td>
<td>4.24 1.66</td>
</tr>
<tr>
<td>2000</td>
<td>2.14 5.26 2.64</td>
<td>6.2 8.08 5.32</td>
<td>3.86 8.45 7.26</td>
<td>9.31 9.42</td>
</tr>
</tbody>
</table>

† F, first 10 d of a month. ‡ M, middle 10 d of a month. § L, last 10 d of a month.

Table 3. Predicted length of growing season and dry matter of Xiang-21 in 1999 and 2000 using Visual Rice Growth Model (VRGM).

<table>
<thead>
<tr>
<th>Year</th>
<th>Length of growing season</th>
<th>Change in length of growing season</th>
<th>Dry matter</th>
<th>Change in dry matter</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>d</td>
<td>%</td>
<td>t ha⁻¹</td>
<td>%</td>
</tr>
<tr>
<td>1999</td>
<td>106</td>
<td>5.67</td>
<td>10.415</td>
<td>11.38</td>
</tr>
<tr>
<td>2000</td>
<td>100</td>
<td>11.60</td>
<td>11.69</td>
<td></td>
</tr>
</tbody>
</table>

SPECIFICATIONS

The VRGM (version 2.5) was developed with Visual C++ version 6.0 (Enterprise Edition). The VRGM source code occupies 500K, and the executable code is 896K. The size of the complete VRGM package is 9.2M. This package contains many image files, dynamic link library (DLL) files, example data files, and a help file. Because VRGM was developed under Windows 95 and

The Use and Performance of Visual Rice Growth Model

To illustrate the use of the VRGM, we used it to investigate the impact of air temperature and solar radiation variations in 1999 and 2000 on ‘Xiang-21’ in Ningxian of Hunan Province, China. Xiang-21, a cultivar of early rice, initially grows under cool conditions, but the vegetative, flowering, and harvesting periods are hot. Table 2 shows mean temperature and sunshine duration of the first, middle, and last 10 d of a month in 1999 and 2000 during the growing season. Irrigation adequately meets the rice’s water requirements, but air temperature and solar radiation can notably influence rice productivity.

In both 1999 and 2000, sowing was performed on 31 March and transplanting on 28 April. The VRGM estimates a longer growing season for Xiang-21 in 1999 than in 2000 (Table 3), which corresponds to the temperature variations during the 2 yr. The input data needed for VRGM are the same as RSSM model and are listed in Table 2. Figure 8 also shows the effect of temperature variations on tillering and leaf expansion predicted by VRGM. But late in the 1999 growing season, a sunshine duration shorter by 2 to 6 h (Table 2) resulted in less photosynthesis and lower dry matter accumulation (Table 3). The simulation results of VRGM match the measured data well, e.g., the difference between the simulation data and the measured data for the length of growing season is 2 to 4 d, and mean correlation coefficient between simulation data and the measured data for the leaf age is 0.9525. These simulations of Xiang-21 were saved as example data in two projects in VRGM package.
98, it has optimal performance when running under Windows 95 or 98. The VRGM can also run under Window 2000/NT, but its GUI face usually does not show well. The user can enter weather and agronomic data directly from spreadsheets. A setup wizard created with InstallShield 6.0 conveniently installs the VRGM package from compressed files. The user’s guide is included with the VRGM package so that users can learn VRGM quickly. The VRGM version 2.5 is available free at http://ricemodel.nease.net/ (verified 30 June 2003). The contact person is the corresponding author of this paper.

**DISCUSSION AND CONCLUSIONS**

Object-oriented programming provides a framework within which to resolve a crop simulation system into many subsystems, allowing a modeler to create components or objects that simulate the behavior of these subsystems. Instead of having to deal with the whole system at once, the modeler can individually implement small parts of the system. At the same time, OOP implicitly narrows the definition of a crop model to cropspecific processes such as growth and development. This excludes other processes that are often assumed to be part of crop models, such as weather and the crop planting decision (e.g., Spitters et al., 1989; Van Evert and Campbell, 1994).

The MDV paradigm, an extension of OOP in crop modeling, provides the modeler with a flexible scheme of duty division for classes and system management for crop modeling software. How to organize the objects in a crop modeling system into a perfect system while avoiding narrowness is a problem for further study.

The overall objective of this research was to define and implement a modeling framework, MDV architecture, that can be used in all simulation programming with OOP. In this paper, the document-view architecture in MFC is extended to MDV architecture for agricultural modeling systems. In MDV architecture, documents act as intermediaries between simulation models and user interfaces. Views get data from documents and show the results to users. The modeling system can be integrated into several MDV groups, such as one each for weather and crop. Different MDV groups communicate efficiently with each other only through documents. The system structure is clear and easy to administer, and communication among classes is more efficient. The modeler can add classes or a MDV group to extend the system without having to change system structure.

The VRGM is presented here as an example to illustrate the implementation of the MDV paradigm. In VRGM, the classes are grouped into three MDV groups. Other similar MDV system structures could produce the same results as described here. The data directory was adopted to manage views in the data area and to coordinate MDV groups in the VRGM system. With careful design, the simulation system can both give crop management decision and avoid the aforementioned narrowness. As an extension of OOP programming in crop modeling, the main advantage of MDV architecture is that the architecture separates the domain model, data management, and user interface. The modeler can integrate a modeling system into several MDV groups, which allows for the development of sophisticated modeling software.

**ACKNOWLEDGMENTS**

The authors thank Associate Editor Robert P. Ewing and three anonymous reviewers for their valuable comments as well as help with the language. This study was funded by the National Rice Scientific Project, part of the “National Ninth Five-Year-Plan,” and the Scientific Foundation Project of Hunan Province (01JJY2033).

**REFERENCES**


Zou, Y., Q. Tang, C. Hu, S. Liu, and D. Xiao. 1991a. Dynamic simula-

---

Statement of Ethics

American Society of Agronomy

Members of the American Society of Agronomy acknowledge that they are scientifically and professionally involved with the interdependence of natural, social, and technological systems. They are dedicated to the acquisition and dissemination of knowledge that advances the sciences and professions involving plants, soils, and their environment.

In an effort to promote the highest quality of scientific and professional conduct among its members, the American Society of Agronomy endorses the following guiding principles, which represent basic scientific and professional values of our profession.

Members shall:

1. Uphold the highest standards of scientific investigation and professional comportment, and an uncompromising commitment to the advancement of knowledge.
2. Honor the rights and accomplishments of others and properly credit the work and ideas of others.
3. Strive to avoid conflicts of interest.
4. Demonstrate social responsibility in scientific and professional practice, by considering whom their scientific and professional activities benefit, and whom they neglect.
5. Provide honest and impartial advice on subjects about which they are informed and qualified.
6. As mentors of the next generation of scientific and professional leaders, strive to instill these ethical standards in students at all educational levels.

Approved by the ASA Board of Directors, 1 Nov. 1992