Application Frameworks, Information Systems, and Intermodal Surface Transportation Efficiency Act

DANIEL S. HALBACH

Numerous problems traditionally exist in translating engineering models into usable software systems. The high-level logic of the model is often compromised or confused by the low-level programming logic. The Intermodal Surface Transportation Efficiency Act of 1991 (ISTEA) has established the need for six information management systems, the requirements for which may serve to magnify these problems. An approach to solving these problems involving application frameworks is presented. This approach promises to empower engineers and decision makers by relieving the constraints imposed by traditional software development practices. Application frameworks appropriately place the emphasis of software system development on engineering modeling rather than on details of programming. A 6-year U.S. Department of Defense logistics management program that provides evidence of the viability of application frameworks is described. General recommendations for ISTEA information systems are provided.

Translating engineering analysis and decision models into information systems and decision support software has traditionally been subject to several common problems:

- The information in the models is often lost or obscured in the resulting code;
- The structure and complexity of the models are sometimes compromised to facilitate simpler code structure and programming logic;
- Low-level control logic and language-specific overhead are intermixed with the higher-level logic of the original model, making it difficult to locate the model in the code; and
- The resulting code is often difficult to understand, maintain, reuse, and tailor.

Although these problems are perhaps tolerable when the system is developed and used by the same engineers or decision makers in a limited setting, problems such as these are magnified when the system has requirements and expectations that exist for the management systems defined by the Intermodal Surface Transportation Efficiency Act of 1991 (ISTEA) (*1*). Specifically, ISTEA establishes six management systems: pavements, bridges, congestion, highway safety, public facilities, and intermodal facilities. The requirements and expectations for these six systems include

- A set of "procedures, within the State's organizations, for coordination of development, establishment, and implementation of the management systems";
- The ability to tailor the systems to "meet State, regional, and local goals, policies, and resources, [while remaining] acceptable to the Federal agencies";
- The "use of data bases with a common or coordinated reference system and methods for data sharing";
- The need for "documentation that describes each management system . . . for the Federal agencies to determine if the systems fulfill the [intended] purpose";
- "Outputs (e.g., policies, programs, and projects) [that can be] integrated into the metropolitan planning process"; and
- A method to handle "interrelationships among systems to address outputs and issues related to the purposes of more than one management system."

Although the nature of many of these factors is organizational as well as technical, the six ISTEA systems are intended to help agencies at all levels deal with each of these factors. To do this the management systems must

- Ease, not increase, the burden of analyzing and sharing data;
- Provide a clear mapping to the decision models on which their designs are based;
- Be expressed in a manner that is independent of any particular hardware, operating system, data base, or graphical user interface (GUI) platform; and
- Provide clear, direct support for system evolution as state and federal policies evolve.

In short ISTEA management systems must enable engineers and decision makers in all agencies at all levels to do their jobs effectively and efficiently. Moreover the cost of developing, implementing, maintaining, and using these systems must not outweigh the benefits to be gained from them.

Note that although various sources define the following terms differently, for the purposes of this paper the terms *information system, management system,* and *decision support system* will be used interchangeably.

SOLUTION APPROACH

Application frameworks (*2*) offer an innovative yet reasonable approach to solving the problems addressed above. Although the concept of application frameworks has only recently emerged as a software engineering discipline, application frameworks have been in existence for some time. In fact GUIs such as those defined by MS-Windows and Apple Macintosh and Relational Data Base Manage-

Pareto Associates, Inc., 1290 Bay Dale Drive, Suite 141, Arnold, Md. 21012.

ment Systems (RDBMSs) are examples of application frameworks that have been well received and proven in their respective domains. They are now showing to be equally applicable to other domains, including the scheduling, planning, and decision support required by ISTEA management systems.

Although the GUI and RDBMS examples stated in the preceding paragraph are known by the specific programming languages and "tool kit" libraries that support them, an application framework is fundamentally more than just a language or library of routines. An application framework is a specific, tangible architecture designed to support a particular problem domain. Because they are intended to support entire domains, application frameworks are designed with the goals of tailorability and direct mapping to underlying engineering models. The discipline of application frameworks has evolved in part because of advances in object-oriented programming (OOP), but its concepts and approach can be understood independently of OOP. However, readers are encouraged to learn more about object-oriented software development from the references provided (*3–5*).

Perhaps the best method for describing the concept of application frameworks is via the analogy of a computer circuit board. The board itself provides a tangible, specific architecture designed to allow the computer chips that plug into it to interact in a controlled, predictable manner and to serve a useful purpose. Any particular socket on the circuit board can be occupied by chips from different vendors and with differing characteristics, provided that

- The chip's pins will fit into the socket,
- The chip can receive the full set of inputs sent by the circuit board, and
- The chip will respond with outputs comprehensible by the rest of the circuit.

An application framework is like a circuit board in that it provides the fixed, but generic, architecture for solving a given class of problems. The framework's design provides "sockets" into which specialized software components may be inserted. These specialized components provide the mechanism through which an application framework can be easily tailored while still conforming to the general architecture. The application framework defines the general solution approach to the associated domain of problems, whereas the specialized components tailor the framework to a specific purpose or platform. The specialized components provide an added benefit of encapsulating the low-level details of their implementation. Thus, the framework's representation of the domain model remains clear and separate from these implementation details, adding to the comprehensibility and maintainability of the code.

As stated, constructs provided by OOP (and object-oriented languages, such as C++) directly support the definition and specialization of tailorable components that can be plugged into an application framework. However, approaches exist for creating application frameworks that are independent of object-oriented programming languages (*6,7*). Application frameworks can be tailored for a particular platform (i.e., hardware, operating system, GUI, or data base) simply by creating or refining components to meet the requirements and protocols of the target platform. Likewise, the specific focus and details of the underlying engineering model can be tailored through the same process of defining specialized software components and inserting them into the general framework. The pavement management system (PMS) example in the next section will further illustrate the nature and benefits of application frameworks. The following are keys to a successful application framework:

- It is based on sound engineering analysis and a well-defined domain model;
- It should clearly define and portray the overall goals of the management system and the decision makers who will use it;
- It should clearly define which parts of its structure are tailorable and which are immutable (i.e., which parts of the framework's code are analogous to the replaceable chips and which correspond to the fixed circuit board); and
- It should be developed at an appropriate level of abstraction.

This last point requires further explanation. The level of abstraction of a framework is basically its degree of generality. A highly abstract framework can be applied to a broad range of applications but provides less direct support to any specific application. Conversely a less abstract framework provides more direct support to a specific subset of applications, and is thus easier to implement and tailor for that subset. However, that subset covers a narrow range of applications in comparison with a more abstract framework. The proper level of abstraction is important for two reasons. First, an attempt to instantiate an excessively abstract framework often results in frequent hacking (i.e., opportunistic coding in the absence of design) because of the lack of support from the framework. Second, an attempt to instantiate a framework with an inappropriately low level of abstraction also invites hacking to subvert or circumvent the parts of the framework that do not apply to the application. To address this abstraction issue, application frameworks are designed as a hierarchy of abstractions in which each level in the hierarchy contains frameworks that are specializations of those at the next higher level. For example, an ISTEA-applicable framework might be a specialization of a constraint-based scheduling framework, which in turn is a specialization of a generic constraint-based framework.

The concept of application framework tailorability desires special note. The traditional notion of tailoring involves the end user's ability to define specific values of parameters used in a system's algorithm. For example, the tailoring of existing PMSs is typically a matter of setting break points in a decision tree. Although this type of tailoring is no doubt important, application frameworks offer an additional and more powerful means of tailoring a system via the specialization of components. As shown in the following PMS example, these components can be much more than simple break points. Components can be pavement or bridge classifications, traffic categories, environmental characterizations, and decision algorithms, among others. Each component can define its own specialized features and behavior within the general constructs of the component type. Each pavement type, for instance, can define its own performance curve that will automatically be invoked by the framework at the appropriate time. Thus, new pavement types (with their own performance curves) can be added to an existing pavement management framework without the need to restructure or redesign the rest of the framework. This is true only if the original framework has been appropriately designed to isolate pavement type as a tailorable component (i.e., the framework's circuit board provides a socket for pavement type, instead of hardwiring it).

PAVEMENT MANAGEMENT EXAMPLE

Although application frameworks are more than merely a simple flow chart, the algorithm embodied in a flow chart can be an im-

portant distinguishing feature of a framework. Thus, the pavement management example of application frameworks provided in this section begins with the following algorithm:

1. For each time period (e.g., year or season) of the analysis or projection period:
   2. For each pavement section in the inventory:
      3. Determine any changes to traffic or environment (i.e., soil, climate, etc.).
      4. Determine pavement performance during that period (i.e., change in condition).
      5. Recommend treatment based on condition, structure, functional class, traffic, and so on.
   6. Prioritize the recommendations based on condition, structure, functional class, and so on.
   7. Apply treatments as the budget allows.
8. Summarize and report.

From this algorithm the need for the following classes of components can be inferred:

- A time line: an overall analysis period divided into discrete steps (Step 1);
- An inventory of items (e.g., pavements) to be maintained (Step 2);
- Factors (e.g., traffic) that affect the condition of inventory item over time (Step 3);
- Performance prediction functions or algorithms (Step 4);
- Treatments for maintaining and rehabilitating inventory items (Step 5);
- Mechanisms for recommending treatments for inventory items (Step 5);
- Mechanisms for prioritizing the importance of recommendations across the entire inventory (Step 6);
- A means of expressing the costs of treatments, presumably in dollars (Step 7);
- Budget constraints that limit the number and types of treatments applied in a given period (Step 7);
- Relationships between treatments, their applicability to each type of inventory item, and the improvement they make to condition (Step 7); and
- A format for defining and expressing the degree to which the projected treatments achieve the desired goal of the system (Step 8).

Each of the component classes should have a well-defined protocol (i.e., set of functions to perform or responsibilities to carry out). These protocols provide the structure within which a framework can be tailored. As with plugging chips into a circuit board, any component that conforms to the protocol of its generic class (i.e., the socket) can be inserted into the framework. Thus, the protocol defines

- The syntax (name, type, and parameters) of the attributes and functions that apply to the component (i.e., how the chip fits into the socket);
- The context and purpose of each of those attributes and functions (i.e., the inputs the chip is expected to receive from the circuit board); and
- The format, content, and range of the attributes and function return values (i.e., chip outputs that will be comprehensible by the rest of the circuit)

It should be noted that the original algorithm given in this example should also be a tailorable part of the overall framework. This is important for two reasons. First, the algorithm addresses the overall goal of the management system, which should also be tailorable. For example, the original goal—achieving the best network condition given the budget constraints—could be inverted into determining the budget required to achieve a desired average network condition. Second, alternative algorithms could be used to address the same goal. For example, in the algorithm given earlier, Steps 1 and 2 could be reversed, making the primary loop address sections instead of years. Thus, instead of making recommendations across the entire network on an annual basis, the algorithm would, at each step, define an entire life cycle of an individual pavement section (in order of section priority and based on remaining budget). Note that even when the algorithm is changed the definition of the other components (and their protocols) remains the same.

RELATED WORK

Evidence of the validity of the claims made in this paper for the benefits of application frameworks technology has been provided by a 6-year program to provide management systems to the U.S. Department of Defense (DOD) logistics management community (*8*). In the description of this DOD program that follows, the reader will note a remarkable similarity between this domain and the transportation management domain addressed by ISTEA.

The DOD logistics management community encompasses a broad range of subdomains, including funds management, contracts management, personnel planning, inventory and supply tracking, maintenance and modification scheduling, and reliability analysis. Users of these subsystems (i.e., the logistics planners) can be characterized as follows:

- Approximately 1,000 users are spread across 20 sites that operate semi-independently.
- Each site must adhere to common policies established by a central command.
- Information systems are tailorable for each site within these general constraints.
- The systems are used to manage scarce or shared resources (especially time and funding).
- Data must be shared between the multiple applications as well as between sites.
- Each site must provide standardized reporting to the centralized command.
- Coordinated decision making is required for resources managed jointly by multiple sites or multiple applications.
- System requirements are constantly evolving to support a changing environment.

The technical characteristics of the management systems themselves include the following:

- "Legacy" systems must be supported and integrated into the overall system environment.
- All systems must have a GUI to facilitate ease of training and use.
- Data bases may reside across multiple, distributed data base servers.
- Both DOS-based and UNIX-based workstations are supported.

Despite these seemingly overwhelming challenges, the application framework approach described herein has consistently provided low-risk, cost-effective solutions for the DOD logistics management community. On average systems originally budgeted to take 1 year or more to develop are now being delivered in 3 months at a significantly reduced cost (*9*). The generalized logistics applications have been ported and tailored for the various sites with only a fraction of the time and funding typically spent on similarly sized DOD software projects. Specific examples of the efficiencies that applications frameworks have provided for DOD logistics management include the following:

- An automated scheduling system that plans all future modifications to the Air Force fleet of C-130 cargo aircraft was delivered in 5 months, including a graphical editor running in X Window on a UNIX workstation.
- An Army inventory and supply management system that also automates all associated DOD forms was developed and delivered in 3 months.
- A funds management system that plans and tracks sources, commitments, and expenditures of six categories of Air Force maintenance, labor, and materials funding was produced and delivered in 5 months.
- An Army aircraft reliability analysis system that analyzes performance and failure data to predict future maintenance and support requirements was produced in 3 months.

The requirements for each of these systems to have distributed data bases, multiple site coordination, and graphical reporting resulted in original development schedules that were as much as five times as long as what was actually achieved via the help of frameworks. The successful implementation of application frameworks in the DOD logistics management domain is evidence of their applicability to transportation information management system development because of the nearly one-to-one mapping between the aspects of the two domains. In fact the only significant conceptual difference between the domains is that one deals with aircraft, military bases, on so forth, whereas the other one deals with pavement sections, bridges, and so forth.

GENERAL RECOMMENDATIONS FOR ISTEA INFORMATION SYSTEMS

Perhaps the most promising characteristic of application frameworks is that they provide a clear representation of the underlying engineering model free from the extraneous implementation details that are encapsulated within the components. This takes the emphasis of system development off programming and puts it back on engineering and decision modeling where it belongs. As a result application frameworks can actually clarify models rather than obscure them. Experiences gained in the creation of information systems for logistics management, enhanced by this clarifying nature of application frameworks, are the basis for the recommendations provided in this section.

Decision support is more than just data base management with the ability to provide summary reports. Decision support systems must also predict future conditions to aid planning and scheduling. For this planning portion of the management system, the system inputs comprise six general categories:

- Resources and supplies. These represent expendable commodities that are used as part of maintenance and rehabilitation activities. They are typically associated with a simple unit price for the purposes of budgeting. Examples include labor and materials. Units of time required to perform activities may also be treated as a resource, depending on the nature of the decision model.
- Fixed assets. These are relatively permanent entities, such as maintenance equipment or facilities, that represent long-term, capital investments. Although depreciation of these assets may be considered, routine maintenance costs for these assets are usually not a direct concern of the decision models.
- Inventories. These are collections of entities, such as bridges or pavement sections, that share properties of both fixed assets and expendable resources. Like assets, they have some fixed properties, such as the geometry and initial capital costs of pavement sections, while also having resourcelike expendable attributes, such as deteriorating serviceability. Thus, unlike fixed assets the maintenance cost of the inventory is precisely the concern of the management system.
- Environmental factors. These factors define external inputs that are not under the direct control of decision makers, including political, economic, climate, and traffic considerations.
- Goal definitions. Multiple goals may be addressed during the planning process. Examples include determining the budget required to bring the inventory to a desired level of serviceability and determining the highest achievable level of condition across the inventory within a given budget constraint.
- Decision models. These are the procedures or algorithms that transform inputs into outputs.

These decision models can take on several forms, but in general they share the following common features:

- A cyclic, iterative simulation engine to support such things as state-transition models, life-cycle models, or Markov chains;
- Optimization criteria (e.g., objective functions, heuristics, or decision trees) that prioritize activities and determine which of two candidate solutions or alternatives is preferable; and
- A set of constraints or boundary conditions that define feasible solutions, including budgetary constraints, sequencing and precedence of activities, and the appropriate pairing of treatments to inventory item structures and conditions.

The planning and scheduling process provides the following three general categories of outputs:

- Activity schedule. This is the calendar of activities or treatments to be performed when, by whom, and at what expected cost.
- Constraint summary. This is a listing of each occurrence of a constraint that either imposes an active limit on the schedule or has been violated in the case in which two or more constraints are in conflict.
- Condition summary. This provides a summary of expected inventory condition by category (e.g., functional class, region, and structure and year.

These general categories of inputs, outputs, and decision models can and should be represented in an application framework as tailorable components to provide the consistent, maintainable, comprehensible, and tailorable systems necessary for cost-effective decision support. Experience has shown that the issues that affect the viability of application frameworks for management systems fall into three categories: suitability, certainty, and quality. Each of these is individually addressed below.

Suitability

Suitability refers to issues concerning both the models and their inputs that involve the notion of granularity and interdependence. Granularity, also referred to as scale or resolution, is primarily concerned with the required frequency, accuracy, precision, and volume of input data. The data required to drive the models must be feasibly collectable; otherwise, the models will lose their utility. This is the "garbage in/garbage out" principle. More than any other single cause, modeling efforts fail because of the lack of adequate data.

Model interdependence becomes a significant issue when models share triggers, or thresholds, or jointly affect the condition of the inventory. For example, a pavement section should not receive a seal coat and an overlay in the same time period. Likewise a model that addresses faulting in a jointed pavement should also affect the pavement's serviceability index in a consistent fashion. These example problems may appear obvious, but they have been observed in existing, fielded PMSs. These types of interdependency problems are often hard to detect when models are run across large data bases and produce only summary outputs.

Interdependency issues have been successfully handled in practice through the use of matrix interpolation and event-driven modeling. For example, pavement performance models that are based on traffic loading, not time, can be synchronized with time-based models by producing a matrix of outputs at regular time intervals for assumed levels of traffic. The matrix is then used instead of the original model in the subsequent time-based simulation. Event-driven systems can be used to allow asynchronous models to run concurrently by maintaining a single calendar of events (or triggers) that is used to coordinate the models. It is also frequently necessary to prioritize constraints across models, since situations often arise in which competing constraints conflict in a given decision and one must win out. It is useful to provide common units of cost and benefit across models to facilitate the arbitration of conflicting constraints. In any case constraints should be defined as explicitly as possible, even when they are inherently embedded in a decision algorithm.

Certainty

Issues of certainty in models are introduced primarily in the form of stochastic processes and accuracy/precision trade-offs. The stochastic, or probabilistic, nature of physical and economic systems is often ignored in practice with potentially disastrous results. When mean time between failure (MTBF) metrics and interest rates are treated as absolutes, significant risk factors are completely overlooked. Accuracy/precision trade-offs become problems when modelers assume that data provided with several digits of precision are actually accurate to the last decimal place. Both types of certainty issues have been successfully handled in practice via Monte Carlo simulation whereby what-if scenarios are run in batches, spanning likely input ranges.

Quality

Issues of quality are usually characterized by a number of trade-offs, including efficiency versus effectiveness, risk versus return, and constraints versus penalties. Efficiency is usually defined in terms of resource utilization, whereas effectiveness is measured in terms of goal attainment. Management systems for which the effectiveness goals have not been clearly defined frequently overemphasize the locally efficient use of a particular resource at the expense of a more globally effective solution. As mentioned previously risk reduction (i.e., uncertainty management) is frequently ignored as an overall system goal. As a result a solution that provides a high expected return may be chosen, even though risks inherent in the solution may diminish its feasibility, whereas a more robust solution with a slightly lower expected value may be preferable. Reductions in data collection costs and system response time are examples of other goals that are often slighted in the model design.

The notion of system quality must also include user considerations, such as ease of use and comprehensibility. In an effort to create academically or theoretically sophisticated models, model developers have shown a tendency to ignore these user considerations. User support includes the use of GUIs, but it goes beyond that. For example, a highly optimized model that produces drastic changes in the output for a relatively minor change to the input data is inadequate as a decision support tool, despite its optimization, because it cannot be easily comprehended.

Similarly models that rigidly enforce constraints, although they are theoretically sound, can be difficult to use in real environments. For example, because of political or other external factors, decision makers are often faced with situations in which a particular maintenance activity or rehabilitation project must be dropped into the final plan or schedule, even though the decision model has not chosen the activity for the particular time slot. If the system does not allow these unexpected drop-ins because they are theoretically infeasible, the system will be of little direct use to the decision maker. The system must model the real environment, not the ideal one, which means that budgets may overrun and schedules may be rearranged. In practice rigid constraints can be replaced by "soft" constraints that include a cost, or penalty, for violating the constraint. In this manner exceptions to constraints can exist, but at a cost. In the simplest case this cost may just be a warning issued to the user that an infeasible situation has occurred. As previously stated a means of prioritizing constraints and common units of cost and benefit are useful in these situations.

General Useful Features

Identifying the problems and designing the solutions presented in this section are examples of an "easier said than done" situation. To aid in problem identification and solution design, the following list describes the features that have been found to be useful in information systems in general:

- An ad hoc querying capability to create specialized reports;
- What-if scenario support (i.e., a baseline plan and variations on the baseline);
- An ability to create a representative subset of the inventory data on which scenarios may be tested;
- Assertions and exception handling to identify anomalies and debug constraints and algorithms;
- Support for drop-in treatments, that is, user-specified activities that override the model's recommendations;
- Graphical reports to aid in visualizing results; and
- Graceful fault handling when one of a pair of conflicting constraints must be violated.

CONCLUSIONS

To readers who have survived the various software fads and "snake oil" salesmen over the past decades, application frameworks may sound like the next in a long line of would-be panaceas. In truth the benefits of frameworks have been proven in the DOD logistics management domain and are based on sound software engineering principles that have held up in practice. No software development approach will ever remove the need for thoughtful planning and design; in fact careful modeling is the foundation of application framework development. Frameworks can, however, make the modeling and decision-making processes more efficient and effective by removing extraneous programming overhead. Application frameworks appropriately place the emphasis of software system development on the engineering and decision modeling rather than on coding, thereby empowering engineers and decision makers by relieving the constraints imposed by traditional software development practices. The benefits of increased portability, tailorability, and maintainability are more readily apparent, but they should not overshadow the less tangible benefits that frameworks have for supporting intuitive domain modeling.

Although any popularity that application frameworks might gain in the transportation industry will no doubt result in a certain amount of marketing hype and snake oil salesmanship, the foundations for deriving real, sustainable benefits are already in place. Numerous commercial vendors already exist for the related object-oriented technology (e.g., C++ compiler vendors), and other government and commercial arenas have already embraced the concepts. The approach is based on academically sound principles and is independent of any commercial product, platform, or programming environment. Thus, frameworks can prove to be a viable and well-received technology for transportation information management system development in the years to come.

REFERENCES

1. Title 23 CFR, The Intermodal Surface Transportation Efficiency Act of 1991. *Federal Register,* Vol. 58, No. 39, Part 500, p. 12115, March 2, 1993.
2. Johnson, R. Documenting Frameworks with Patterns. *Proc., OOPSLA 1992, ACM SIGPLAN Notices,* Vol. 27, No. 10, Oct. 1992, pp. 63–76.
3. Booch, G. *Object-Oriented Design with Applications.* Benjamin/Cummings Publishing Co., New York, 1991.
4. Coad, P., and E., Yourdon. *Object-Oriented Analysis,* 2nd ed. Yourdon Press, Englewood Cliffs, N.J., 1991.
5. Rumbaugh, J., et al. *Object-Oriented Modeling and Design.* Prentice-Hall, New York, 1991.
6. *Common Object Request Broker Architecture (CORBA) Interface Definition Language (IDL).* Object Management Group, Framingham, Mass., 1993.
7. Johnson, R., and J. Zweig. Delegation in C++. *Journal of Object-Oriented Programming,* Nov. 1991.
8. Special Operations Logistics Automated Management System (SLAMS). Contract F09603-86-G-0054. ARINC Research Corporation.
9. Systems Strategy: Object-Oriented Programming. *Washington Technology,* Vol. 8, No. 9, Aug. 12, 1993.

Publication of this paper sponsored by Committee on Computer Technology.