Using Shlaer-Mellor For Client/Server Solutions

by Sally Shlaer and Stephen J. Mellor

Project Technology, Inc.
10940 Bigge Street
San Leandro, CA 94577-1123
http://www.projtech.com
tel: 510 567-0255
tel: 800 845-1489

January, 1996

© Copyright 1996 by Project Technology, Inc. All rights reserved.

Introduction

One of the key issues facing IS groups today is the move from centralized, mainframe-based computing toward distributed, network-based architectures. Today's client/server systems are characterized by heterogeneous computing platforms, networking protocols, distribution of data, processing and applications. The rapid move toward client/server systems has led to significant changes in the way Information Technology (IT) systems are developed and deployed. While there are a number of important user benefits to be gained by moving to client/server architectures — most notably easy access to data, cost-effective computing platforms and improved flexibility for responding to changing business conditions — the transition does not come without challenges.

Initial experiences have shown that the move toward client/server systems can be both risky and difficult. The challenges of deploying these systems include the need to maintain control over the development process and an increased system complexity caused by more sophisticated data processing and storage models. For example, it is difficult to predict and optimize the performance of network-based systems. Systems designed to meet the needs of a single workgroup or department too often scale up poorly when expanded to meet enterprise-wide demands. Additionally, as underlying processing and networking technologies continue to evolve, application developers are required to create systems that can continue to adapt to these new technologies.

Organizations that have expertise in developing applications based on mainframe technology (such as COBOL and SNA) often lack experience with the technologies required for client/server development. While these organizations may have a clear understanding of the application requirements, the

new technologies pose challenges. This results in today's implementation technologies (such as TCP/IP, C++ and distributed databases) having a substantial impact on application development both in terms of cost and of learning curve time.

Client/server implementations are often plagued by:

- Long development cycle times
- Systems that fail to meet performance requirements or do not scale up well
- Confusion over who controls the computing data, resources and development process

Traditional development approaches are unable to solve these problems. A new development process is needed to provide formality and standardization to meet the growing demand for software quality, to address system complexity issues, and to meet enterprise-wide performance requirements. One approach has emerged that meets these demands. Object-oriented analysis and design (described in Appendix A), which has been meeting the challenging requirements of industrial and commercial system developers for a number of years, is now being successfully deployed on client/server information system design.

This paper explores using an object-oriented software development method and the benefits an IS developer can expect to gain from adopting this approach. It examines a popular development method, the Shlaer-Mellor Method (developed by Project Technology, Inc.), and describes how to apply it to client/server projects.

Shlaer-Mellor Method and Client/Server Systems

The Shlaer-Mellor Method provides substantial benefits to the IS application developer (see Appendix B for a detailed description of this method). This method is an innovative approach that addresses many of the challenges inherent to client/server systems development. Specifically, the developer can expect to gain a number of benefits, including:

- The ability to address system complexity issues
- Standardization through a repeatable and manageable development process
- Consistent software quality
- Increased developer productivity and reduced development cycle times

System Complexity

Keys to maintaining the development of a complex system include partitioning the work into doable tasks, and then assigning these tasks to developers with the appropriate knowledge and experience. A major strength of the Shlaer-Mellor Method is its ability to manage large-scale projects that utilize multiple programmers and hardware designers. Using this method, developers partition a system into well-defined conceptual layers, or "domains." For an additional layer of control, domains can be further divided into subsystems: small work units assigned to small teams (i.e., two to four developers) in a divide-and-conquer approach. This results in a better structure for project planning and

© Copyright 1996 by Project Technology, Inc. All rights reserved.

provides a context for assigning different aspects of the system to small, technically-focused development teams.

The domains logically separate implementation issues (such as the underlying computing environment and data storage and processing distribution schemes) from application and user interface issues. All domains can be worked on separately and concurrently, allowing the system developer to explore alternative implementation approaches without impacting the application layer. This ability is particularly important given the complex nature of distributed systems. Too often, it is unclear until late into the development process how well a system will perform under varying loads and how well it will scale up with increased usage. With the Shlaer-Mellor Method, the developer can test different implementation schemes and analyze their impact on performance without delaying application level development. In addition, changes in one domain are isolated from the rest of the system, minimizing any ripple effects.

For organizations making the transition to client/server systems, the divide and conquer approach is particularly powerful. Application programmers are often experienced in COBOL and the mainframe environment but lack expertise with client/server technology (such as TCP/IP, IPX, GUI design and UNIX database systems). With the Shlaer-Mellor Method, project leaders can leverage the applications-layer experience of these programmers by assigning them to the application domain and subsystems. Equally, the new generation of network experts, who may not possess broad application area experience, can be assigned to a subsystem within the implementation domain.

Standardized Process

One of the primary concerns regarding the migration toward client/server systems and the movement of data and applications from the data center into field and department-level operations is the lack of control over the development process. While business managers have a solid understanding of the business problem to be addressed, they typically lack experience in building, documenting and maintaining a client/server based solution. As a result, many organizations have adopted a strategy of combining the skill and experience of the IS department to develop and maintain systems with the business department's ability to define the need for efficient and flexible solutions.

The Shlaer-Mellor Method, with its rich and well-defined process for specifying and modeling systems, can play an active role in bringing the two groups together. Information modeling provides a common language for communicating requirements and design alternatives. The method-defined work products can be used to communicate among development groups, department managers and upper management. Moreover, because the Shlaer-Mellor Method defines the procedure for applying the method to a particular system, it is possible for an organization to use the method for controlling the entire development process.

One of the most overlooked aspects of managing a development project is the difficulty in maintaining up-to-date project documentation during the entire product life-cycle, particularly as a project moves toward completion. Documentation often falls behind as final changes are made to the current product and as development efforts for future versions are initiated. The resultant inaccurate documentation impacts the efforts of both the team maintaining the system and other groups assigned to developing future versions of the system.

© Copyright 1996 by Project Technology, Inc. All rights reserved.

When using the Shlaer-Mellor Method, the pre-defined work products (models) provide complete, accurate and on-going support documentation. Because there is a direct correlation between the models and the resulting code, documentation maintenance enforcement mechanisms are inherently in place. If the code does not match the model, it does not run. As a result, organizations can rely upon correct, up-to-date documentation for all future development and maintenance activities.

To help ensure standardization, training and consulting services can be obtained from an outside organization with extensive experience in OOA methods and real-world implementation issues; this can significantly reduce the learning curve time and help institutionalize the development process. In addition, experienced and trained developers deployed on the different domains within a project can serve as leaders for the rest of the team, providing consistency within the group.

Increased System Quality

One of the driving factors for adopting a standardized development process is the demand for improved quality. Advanced software development methods bring a number of significant benefits to the product development process, including:

- Improved product stability
- Simplified product modification
- Reduced test time
- Better product quality in the field

One way in which the Shlaer-Mellor Method provides the above benefits is by removing defects early in the development cycle. Because the method's analysis models are executable, developers can begin to test system behavior very early in the development process. Using simulation, designers can actually test the analysis models for correct behavior. As a result, the system is built on a well-structured base, reducing the likelihood of system problems — either upon initial deployment or during ongoing, long-term enhancement.

Leveraging the concept of an executable analysis model, the Shlaer-Mellor Method enables system designers to generate application code automatically. The result is less handcrafting of code and a consistent level of quality. Not only does high quality application code avoid prolonged testing phases, it also reduces integration time.

When using traditional methods, significant portions of the system code are actually developed during the test phase. The result is a protracted development cycle in which the test and integration team is forced to create new features and address performance and implementation issues, all in the guise of "integration." With the Shlaer-Mellor Method, the test process is significantly reduced to the point where testing becomes a simple task of certification of the OOA models and the generated code.

© Copyright 1996 by Project Technology, Inc. All rights reserved.

Enhanced Productivity

One of the promises of object-oriented development is the reduced development times obtained through the reuse of software components. The Shlaer-Mellor Method delivers on this promise by logically partitioning the system. Through the effective analysis of separate domains, an organization can create analysis models that can easily be moved from one system to another. An implementation domain that captures an organization's standards for computing platforms, networking and systems software can be moved from one application to another with little or no modification. Higher level domains, such as a user interface or system reporting function, can easily be used on all systems on an enterprise-wide basis. Because the analysis models developed using the Shlaer-Mellor Method are complete and tested through simulation, the task of moving software components to a new system is vastly simplified.

An additional benefit of the Shlaer-Mellor approach is the efficiency gained through concurrent development of both the analysis and design solutions. Unlike the traditional software development technique of sequential analysis, design and implementation, concurrent development allows developers to move forward on all aspects of the system at the same time. The result is a more effective utilization of engineering resources and reduced overall development times.

Perhaps the most powerful productivity-enhancing aspect of the Shlaer-Mellor Method is Recursive Design (RD), or Implementation Through Translation. With RD, users are able to convert analysis models into application code using a standard CASE tool suite with a code generation component. These tools extract model components from a CASE tool database, integrate them with the software architecture domain, and then generate production quality application code. In many projects that have used the Shlaer-Mellor Method, a large majority of the application code was automatically generated (in some cases, 100% of the final code was generated).

AT&T: A Success Story

In the highly competitive world of telecommunications, efficiency and productivity are paramount concerns. When AT&T needed to develop an advanced On-Line Support System (OLSS) for entering and processing customer service orders, this meant designing a client-server system based on leading edge multi-processing UNIX systems, an intricate wide area and local area networking scheme, and distributed data storage and processing.

For Gerry Boyd of AT&T, "best of class" also meant managing the complexity and impact of rapidly changing technology when developing the system's sophisticated software system. Mr. Boyd chose the Shlaer-Mellor Method as the process for the new On-Line Support System application development.

"In the past, we used various Structured Analysis/Structured Design (SA/SD) techniques, as well as more informal processes," said Mr. Boyd. "The problem with these approaches is that our applications have grown in both size and complexity, making it increasingly difficult to bring projects to a successful conclusion. One way to solve the complexity problems is to break a system into smaller pieces, allowing developers to only focus on the aspects of the project that are germane to their

© Copyright 1996 by Project Technology, Inc. All rights reserved.

assignment." And, he adds, "Separating the architecture from the application will most certainly give us a re-usable platform. When we build future systems, we will be able to re-use the basic architecture."

By using Recursive Design, AT&T is able to convert analysis models into application code using a standard CASE tool suite with a code generation component or with a custom developed code generator. One of the key benefits of code generation at AT&T is the consistency it has delivered to the process. Because code is generated from analysis models, implementation quality has improved. Or, as Mr. Boyd puts it, "we do not have to police implementation or worry about hacking."

Summary

Client/server systems present new problems for organizations creating software applications using traditional development approaches. This paper has examined the issues an IS shop faces when developing client-server solutions, including long development cycles, inadequate system requirements, and the lack of a standardized development process. This paper has shown that the Shlaer-Mellor Method provides a significant and measurable way of addressing these issues, representing a long-term, positive return on the investment.

Implementing the Shlaer-Mellor Method requires a commitment from upper management to invest in both expert-level training for the developers and the software development tools designed to support this method. Project Technology provides a Technical Report called "OOA in the Real World" that describes how to begin using the Shlaer-Mellor Method. Project Technology provides numerous reports describing how the Shlaer-Mellor Method is being successfully used in various programming environments and the tools that further enhance this approach. Many of these reports are available for free downloading from our web site **(http://www.projtech.com).**

© Copyright 1996 by Project Technology, Inc. All rights reserved.

Appendix A: Object-Oriented Software Development

While the early spotlight of the software community was directed at object-oriented programming (OOP), the more important issue of object-orientation as a development philosophy remained on the collective periphery. The object paradigm must extend beyond the programming phase of the software development process to realize the full benefits of objects. Today, as object technology gains popularity, the use of an object-oriented software development process is becoming the focal point of the industry. The result is a marked increase in the use of well-defined object-oriented software development methods.

To explore object-oriented methods further, it is helpful to consider the way people develop software. Various practitioners approach the task differently, yet a common thread weaves throughout all software development activities. From the solo hobbyist programmer to a multi-person team building the next generation of a securities trading system, all understand the programming problem, a selection of a solution to the problem, and the creation of that solution. This process is also known as analysis, design, and implementation. Since the meanings of these words vary among software professionals, this paper provides the definitions of these words used by Project Technology, Inc., the creator of the Shlaer-Mellor Method.

Analysis is the *study* of a problem. The analysis of a problem concentrates on what must be done and not on how to accomplish it. The analysis effort results in a complete and conclusive description of the problem.

Design is the *selection* of a particular type of implementation solution for a given problem. The results of the design work are specifications and policies that guide the construction of the solution.

Implementation is the *creation* of the chosen solution for the problem. During the implementation phase, the source code is produced according to the design specifications and guidelines.

Although all software constructors (individual and organizational) perform analysis, design, and implementation, far fewer use formal methods, tested techniques, and effective tools to do so. The question is not whether such things are advantageous — they are. The question is how to forge the methods, techniques, and tools into an effective and efficient software development process.

© Copyright 1996 by Project Technology, Inc. All rights reserved.

Appendix B: The Shlaer-Mellor Method

The Shlaer-Mellor Method, developed by Sally Shlaer and Stephen J. Mellor, provides a unique approach to object-oriented development. Although best known for its approach to Object-Oriented Analysis, the Shlaer-Mellor Method defines a *complete* process for the analysis, design, and implementation of software systems. In a recent report (dated October 1994) IDC, a leading market research firm, identified the Shlaer-Mellor Method as the second most widely used object-oriented method and the leading translation oriented method.

The Shlaer-Mellor Method combines Object-Oriented Analysis (OOA) and Recursive Design (RD) techniques in a comprehensive approach to software development. OOA describes the notation, models and the procedures necessary to discover and model objects in the problem space (see Figure 1). RD provides the transition policies and guidelines for turning analysis models into code for the entire system.

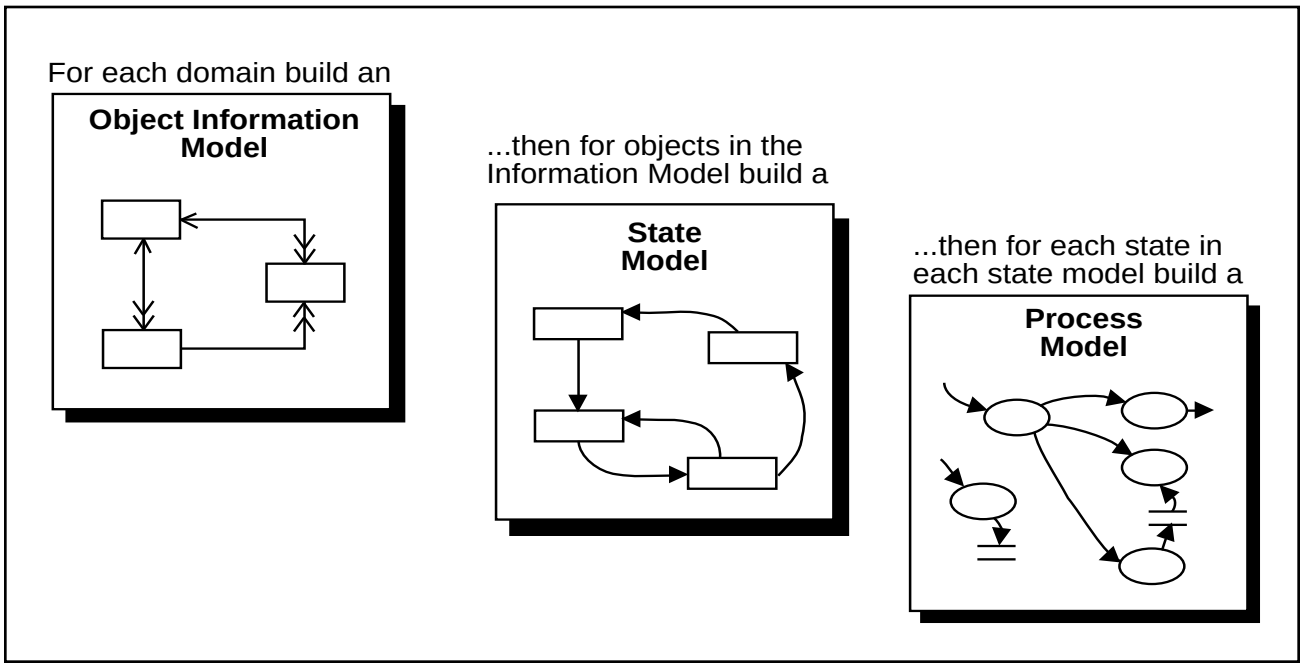

Figure 1. Models used in the Shlaer-Mellor Method

While there are a number of different object-oriented methods in use — and a casual glance at the notations may indicate similarities — there are fundamental differences among the methods. The Shlaer-Mellor Method is unique in that it views analysis and design as two separate problem spaces that can be developed independently and concurrently.

Analysis of the application is represented using the Shlaer-Mellor OOA model and notation while design issues are captured as a software architecture and a set of translation rules. Once these work products are developed, the system designer has sufficient information to translate the OOA models directly into implementation using RD. The implementation may consist of object-oriented code (such as SmallTalk and C++), object-based code (such as Ada), or procedural code (such as C, COBOL or FORTRAN).

© Copyright 1996 by Project Technology, Inc. All rights reserved.

Other object-oriented methods view analysis and design as an entirely sequential process in which the two tasks are separated solely by the level of detail described in the models. With these methods, software development consists of an ill-defined iteration through the phases of analysis, design and code. During each phase, the developers capture additional information as they elaborate on the diagrams until the final elaboration captures the full coding details. While these development strategies take advantage of object-oriented notations, the process they employ has much in common with traditional *ad hoc* development methods.

The Shlaer-Mellor approach features three distinct and unique characteristics of the method:

- Use of domains
- Role of verification
- Implementation through translation

Domains

The Shlaer-Mellor development process relies heavily on the concept of domains. A domain consists of a specific and independently existing subject matter (see Figure 2). A domain is defined as a separate, real, hypothetical, or abstract world inhabited by a distinct set of objects that behave according to rules and policies characteristic of the domain. For example, the abstract world of the user interface has objects such as icons, windows and dialog boxes, each of which has a well-defined behavior — icons can overlap other icons and dialog boxes cannot be moved outside a window. In implementation every system requires a mixture of different subject matters. Typical examples include the application, user interface, operating system, computer language, and reporting domains, as shown in the example in Figure 2.

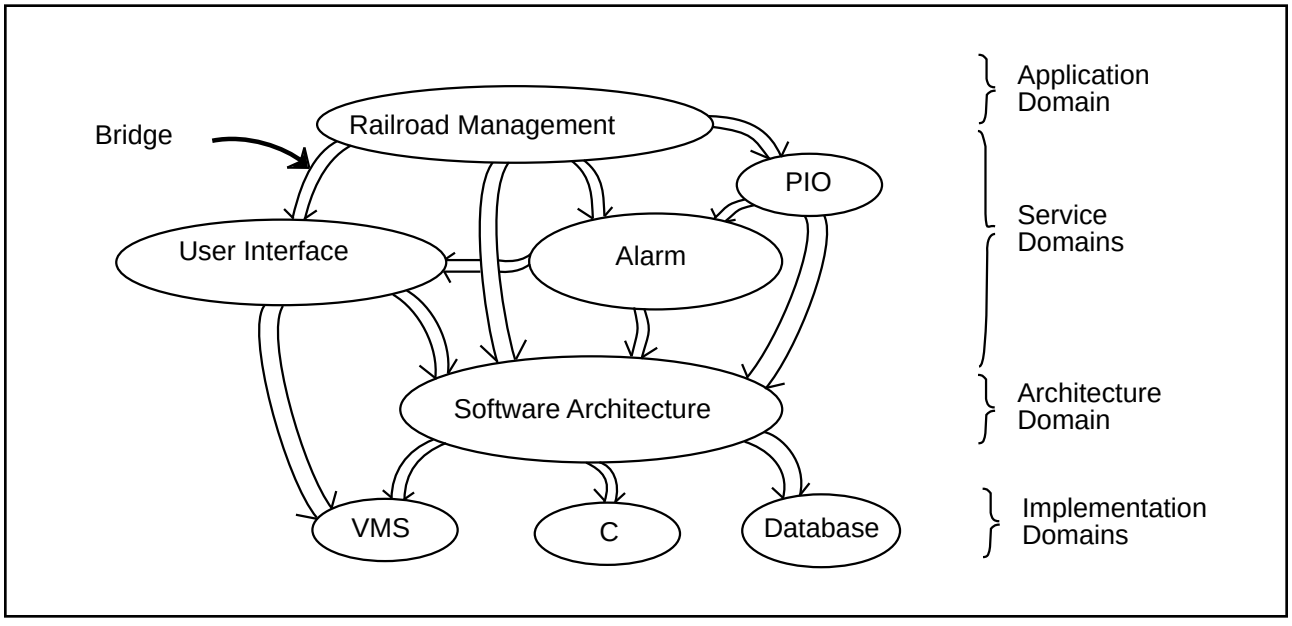

Figure 2. Domains used in a Railroad Management System

© Copyright 1996 by Project Technology, Inc. All rights reserved.

Domains provide separate, reusable frameworks — to be used by the project at hand as well as by other projects that have similar requirements. Through the reuse of entire domains, Shlaer-Mellor Method users often achieve 50% to 60% reusability of the OOA objects as compared to the 5% to 15% typical reusability obtained through other methods. Domains maintain their independence through bridges — structures that connect requirements in client domains to functionality provided by service domains (see Figure 3). It is not uncommon for an organization to develop families of products in which the only difference among the different systems is the application domain.

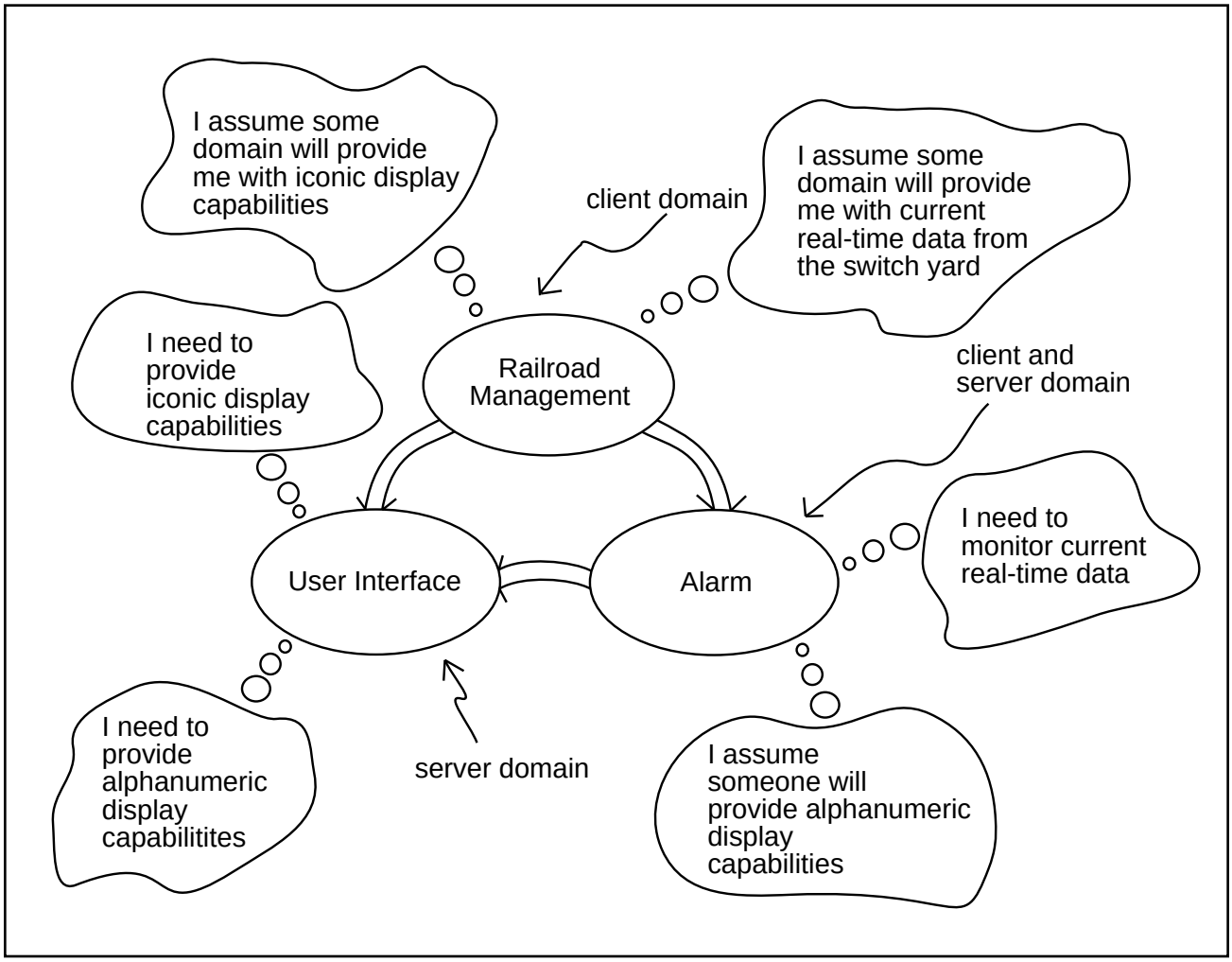

Figure 3. Domains Connected by Bridges in a Railroad Management System

Domains provide two substantial benefits to the Information Technology developer. The first results from the firm separation between the concepts of analysis and implementation and the well-defined mapping between the two. This facilitates better control through the ability to partition large projects among groups or individuals. Methods in the past have tried to separate these two ideas but they often become blurred, or do not provide a defined process to transition between them.

The second benefit is that of allowing a developer to concentrate in his/her area of expertise. For example, an expert in accounting rules can develop the accounts payable domain without having to

© Copyright 1996 by Project Technology, Inc. All rights reserved.

know about GUIs (the User Interface domain) or TCP/IP networks (the Software Architecture domain). Experts in these domains (User Interface, Software Architecture) can be hired and can develop these domains without ever having to learn the company's applications.

Verification

The Shlaer-Mellor Method prescribes various models to represent many ideas within the system. There are rigorous definitions to the semantics of each model and formal rules on how different ideas are captured in the models. These rules and semantics ensure correctness of the information captured. They also allow precise static and dynamic verification of the system.

Static verification is possible because the Shlaer-Mellor Method has an underlying formalism. CASE tools (such as BridgePoint by Project Technology) understand this formalism and can offer the object modeler the point-of-entry data checking. This means that models can only be entered or created correctly.

In the past, most developers were not able to do dynamic checking or verification because the methods were too imprecise to allow direct execution of the models. Developers had to rely on prototypes that did not always match up with the formal system analysis. The Shlaer-Mellor Method allows software developers to have "Virtual Prototypes" driven directly by the OOA models. As soon as a model is developed, it can be executed. A customer can verify its behavior, and then make corrections or modifications directly to the OOA models. This ensures that the final system operates the same way as did the prototypes driven by the same OOA models.

Implementation through Translation

The cornerstone of Recursive Design is the concept of Implementation through Translation. Analysis models are translated directly into application code by applying design rules. The design phase focuses on the selection and creation of the rules. The design rules guide the construction of the solution. During the implementation phase the rules are applied to the analysis models to generate source code.

It is the degree of rigor and precise definition of OOA models that allow the Shlaer-Mellor analysis models to be translated to application code. By using simulation, the correct behavior of the models is verified before the code is generated. In contrast, other methods require the developer to evolve the objects from analysis drawings to final code by adding detail at the various steps in the development process.

The OOA models contain a level of detail that people often associate with detailed design or coding. Considerable time is spent in the analysis phase of a project with much less time spent in the coding phase. In the Shlaer-Mellor Method training courses at Project Technology, Inc., the staff teaches students to expect to spend about 60% of the overall project time developing the OOA models. Overall project development time is not increased, just used differently.

© Copyright 1996 by Project Technology, Inc. All rights reserved.

Sally Shlaer and Stephen J. Mellor

Sally Shlaer and Stephen J. Mellor are the co-founders of Project Technology, Inc. They created the Shlaer-Mellor Method in 1979 while consulting on a large rapid transit system. Together they have written two books on software development: *Object-Oriented Analysis: Modeling the World in Data*, and *Object Lifecycles: Modeling the World in States*, both of which are available from Prentice Hall. They are currently working on a third book in the series entitled *Shlaer-Mellor Method: Recursive Design.*

Project Technology, Inc.

Project Technology's mission is to improve the productivity of real-time software development. As developers of the Shlaer-Mellor Method and BridgePoint automation for software analysis and design, the company has been providing training, consulting, and hands-on implementation services for 10 years. Applications developed using the Shlaer-Mellor Method can be found in defense, telecommunications, financial services, real-time control, instrumentation, manufacturing, transportation and utilities sectors. Customers include AT&T, Abbott Laboratories, Delco, EDS, General Electric, IBM, Motorola, and Westinghouse. Project Technology has offices in several U.S. locations, including Texas, Arizona, and California. They also a U.K. office in Glasgow, Scotland and distributors worldwide.

© Copyright 1996 by Project Technology, Inc. All rights reserved.