Appendix B

Experiences Using the Shlaer-Mellor Method

This is a report prepared after an interview, by the author in November 1993, with those at IBM WSDL who were considering the prospect of adopting the Shlaer-Mellor object-oriented analysis and design method.

The Shlaer-Mellor object-oriented analysis and design method [SM88, SM92] is currently being investigated at the IBM WSDL (Warwick Software Development Laboratory) [DSWW93] and other IBM sites as a means of improving their software quality and productivity. The method is being used in a project to improve a report generator. This main project has spawned two mini-projects. The first project followed the Shlaer-Mellor approach to produce inefficient yet working C code. The second project is still in progress, aiming to build on the work of the previous project by reusing the analysis work done, using a multitasking software architecture and automatically generating C++ code.

The Shlaer-Mellor method is being considered at the IBM WSDL as a replacement for the traditional in-house SD approach based on the use of work-books. The work-books are used by software developers to specify the software for a system, then passed to programmers who code the software in C or the IBM Application System Language (ASL). Programming in ASL involves composing application modules

in accordance with their Application Program Interfaces (API). The existing version of the report generator was developed using the in-house approach and written in ASL.

The sections that follow the overview of the Shlaer-Mellor method give an account of what was learned about software development using the Shlaer-Mellor method in the IBM WSDL through talking with those actively involved at both the technical and managerial levels.

B.1 Overview of the Shlaer-Mellor method

Shlaer and Mellor [SM88, SM92] developed their object-oriented analysis method over the course of several years of consulting practice in information modelling [Mar90]. Although information modelling forms the foundation of the method it also draws from conventional object-oriented analysis methods that model the behaviour and function of systems. The Shlaer-Mellor object-oriented analysis method has the following sequence of stages:

1. Large problems are decomposed into conceptually distinct domains. Four main types of domains are identified in the method: application, service, architectural and implementation domains. Bridges link domains together.

2. The software developer follows domain analysis by constructing an information model or entity-relationship diagram (ERD). The information model consists of objects classes and their attributes with inheritance and aggregation relations defined between them.

3. The software developer defines lifecycles for the objects and relations as state models consisting of states, events, transitions and actions. During this stage the software developer defines timers and other mechanisms for managing concurrent behaviour.

4. An action data-flow diagram (ADFD) is defined for each action in the state model. Actions consist of four types of process: data transformation, data access, data testing and event generation.

In their second book [SM92] Shlaer and Mellor extend their method from the problem domain into the solution domain by describing a transformation from the products of object-oriented analysis to the products of object-oriented design.

B.2 Domain analysis

The following was discovered about domain analysis by talking to those responsible for constructing the end-user-interface for the new application:

- It is possible to determine the client-server relations between modules in the existing system by analyzing their interface definitions and the configuration of modules.
- The service domain consists of all those modules which provide a service in the existing system and the application domain consists of all those modules which are clients in the existing system.
- Client-server relations between modules in the existing system map onto bridges between the application and service domains and bridges between subsystems within domains.
- It is difficult to define bridges to the implementation domain because it is a domain that has yet to be defined. The IBM WSDL is investigating ways of defining a "wrapper" interface for domains which have yet to be defined based on APIs.

These findings suggest that the Shlaer-Mellor method of domain analysis is suitable so long as the system being analyzed is already divided into parts that correspond to domains linked by bridges. In the absence of such a correspondence, domain analysis becomes difficult. This is exacerbated when domains have yet to be defined.

B.3 Using the Teamwork tool in analysis and design

The following was discovered about the use of the Teamwork computer-aided software engineering tool for analysis and design by talking to the various people who

had used the tool:

- The tool is configured to support the notations and principles of the Shlaer-Mellor method. The tool can also be configured for other object-oriented analysis and design methods.
- The most useful feature of the tool was found to be the repository of the products of analysis and design that can be accessed across a network by software developers and provides the source for automatic code generation (Section B.4).
- Developers mainly use the tool for drawing diagrams. The other facilities were found of little use and even as a drawing tool it has its limitations with no intelligent text entry or diagram reformatting.
- The representations of models tend to be much larger than the screen. Developers tend to print parts of the model onto pieces of paper and stick them together to see more than the screen can show.

The above findings suggest that the ability of the tools to hold and distribute information was more useful to the software developers than its ability to process or represent information. A more generalized tool or suite of tools that supports access to a knowledge base and diagramming may have been more appropriate.

B.4 Automatic code generation

The following was discovered about code generation by talking to the person responsible for building the system that performed the automatic conversion of models into code:

- The software architecture for the new system is defined as object class definitions which are reusable. The object classes include the "engine" that drives the state machine and generalized classes for the states, transitions, etc.
- Code is automatically generated by the tool in the following sequence of steps:

1. The information and state models constructed during analysis are retrieved from the Teamwork repository (Section B.3).
2. A skeleton object class definition is generated for each application object in the information model consisting of class, attribute and action names.
3. Specialized state, event and transition object class definitions are generated for each object state model.

Programmers complete the skeleton object class definitions by writing code to implement the actions.

- The generated code consists of an object class definition for each application object, state, event and transition represented during analysis resulting in a large number of definitions.

Automatic generation of code was a feature that attracted IBM WSDL to the Shlaer-Mellor method. However, the findings suggest the generated code is unmanageable because of its complexity. This makes it difficult to test and maintain code.

B.5 Testing

The following was discovered about testing by talking to the various people who were responsible for devising means for checking the quality of code:

- Testing is performed on the products of analysis because they tend to be more structured than the code.
- Tools are required to help understand the complex behaviour resulting from the simulation of multiple state machines acting concurrently. The decision was made not to use the IBM Tuscon Object Oriented Analysis Simulator (TOOAS). Instead, a role-based method is being used at the IBM WSDL in which objects are isolated in turn and their role in the behaviour of the system examined.
- The tools help the tester to visualize the dynamics described in the model and it automatically checks that constraints defined by the tester are being satisfied.

These findings suggest that products of analysis are difficult to understand without the help of powerful simulation tools. They also suggest that the products of analysis define the same behaviour as the generated code. This is probably due to the automatic code generation.

B.6 Conclusion

The Shlaer-Mellor method is far more prescriptive than the traditional approach of using work-books. This has the advantage of tool support and powerful techniques for analysis. However, some of the features of tools and techniques were found by software developers to be too limited. Developers found that the method could only be applied when there was a direct correspondence between the system being analyzed and the notations and principles of the method. Some software developers voiced their reservations about the method and its lack of support for using their experience and knowledge of developing software.