6.1. Goals

Commercial requirements differ from military requirements in their emphasis on preserving data integrity. Lipner [636] identifies five requirements:

1. Users will not write their own programs, but will use existing production programs and databases.
2. Programmers will develop and test programs on a nonproduction system; if they need access to actual data, they will be given production data via a special process, but will use it on their development system.
3. A special process must be followed to install a program from the development system onto the production system.
4. The special process in requirement 3 must be controlled and audited.
5. The managers and auditors must have access to both the system state and the system logs that are generated.

These requirements suggest several principles of operation.

First comes **separation of duty**. The principle of separation of duty states that if two or more steps are required to perform a critical function, at least two different people should perform the steps. Moving a program from the development system to the production system is an example of a critical function. Suppose one of the application programmers made an invalid assumption while developing the program. Part of the installation procedure is for the installer to certify that the program works "correctly," that is, as required. The error is more likely to be caught if the installer is a different person (or set of people) than the developer. Similarly, if the developer wishes to subvert the production data with a corrupt program, the certifier either must not detect the code to do the corruption, or must be in league with the developer.

Next comes **separation of function**. Developers do not develop new programs on production systems because of the potential threat to production data. Similarly, the developers do not process production data on the development systems. Depending on the sensitivity of the data, the developers and testers may receive sanitized production data. Further, the development environment must be as similar as possible to the actual production environment.

Last comes **auditing**. Commercial systems emphasize recovery and accountability. Auditing is the process of analyzing systems to determine what actions took place and who performed them. Hence, commercial systems must allow extensive auditing and thus have extensive logging (the basis for most auditing). Logging and auditing are especially important when programs move from the development system to the production system, since the integrity mechanisms typically do not constrain the certifier. Auditing is, in many senses, external to the model.

Even when disclosure is at issue, the needs of a commercial environment differ from those of a military environment. In a military environment, clearance to access specific categories and security levels brings the ability to access information in those compartments. Commercial firms rarely grant access on the basis of

"clearance"; if a particular individual needs to know specific information, he or she will be given it. While this can be modeled using the Bell-LaPadula Model, it requires a large number of categories and security levels, increasing the complexity of the modeling. More difficult is the issue of controlling this proliferation of categories and security levels. In a military environment, creation of security levels and categories is centralized. In commercial firms, this creation would usually be decentralized. The former allows tight control on the number of compartments, whereas the latter allows no such control.

More insidious is the problem of information aggregation. Commercial firms usually allow a limited amount of (innocuous) information to become public, but keep a large amount of (sensitive) information confidential. By aggregating the innocuous information, one can often deduce much sensitive information. Preventing this requires the model to track what questions have been asked, and this complicates the model enormously. Certainly the Bell-LaPadula Model lacks this ability.