Protocols to Validate Laboratory Information Systems; Proposed Guideline

This proposed document is published for wide and thorough review in the accelerated Clinical and Laboratory Standards Institute consensus-review process. The document will undergo concurrent consensus review, Board review, and delegate voting (i.e., candidate for advancement) for 90 days.

Please send your comments on scope, approach, and technical and editorial content to the Executive Offices.

Comment period ends
12 April 2005

The subcommittee responsible for this document will assess all comments received by the end of the comment period. Based on this assessment, a new version of the document will be issued. Readers are encouraged to send their comments to the Clinical and Laboratory Standards Institute Executive Offices, 940 West Valley Road, Suite 1400, Wayne, PA 19087-1898 USA; Fax: +610.688.0700, or to the following e-mail address: standard@clsi.org

COMMENT

This document provides guidance for developing a protocol for validation of the Laboratory Information System (LIS) as well as protocols for assessing the dependability of the LIS when storing, retrieving, and transmitting data.

A guideline for global application developed through the Clinical and Laboratory Standards Institute consensus process.
The Clinical and Laboratory Standards Institute (CLSI) (formerly NCCLS) is an international, interdisciplinary, nonprofit, standards-developing, and educational organization that promotes the development and use of voluntary consensus standards and guidelines within the healthcare community. It is recognized worldwide for the application of its unique consensus process in the development of standards and guidelines for patient testing and related healthcare issues. Our process is based on the principle that consensus is an effective and cost-effective way to improve patient testing and healthcare services.

In addition to developing and promoting the use of voluntary consensus standards and guidelines, we provide an open and unbiased forum to address critical issues affecting the quality of patient testing and health care.

PUBLICATIONS
A document is published as a standard, guideline, or committee report.

Standard  A document developed through the consensus process that clearly identifies specific, essential requirements for materials, methods, or practices for use in an unmodified form. A standard may, in addition, contain discretionary elements, which are clearly identified.

Guideline  A document developed through the consensus process describing criteria for a general operating practice, procedure, or material for voluntary use. A guideline may be used as written or modified by the user to fit specific needs.

Report  A document that has not been subjected to consensus review and is released by the Board of Directors.

CONSENSUS PROCESS
The CLSI voluntary consensus process is a protocol establishing formal criteria for:

- the authorization of a project
- the development and open review of documents
- the revision of documents in response to comments by users
- the acceptance of a document as a consensus standard or guideline.

Most documents are subject to two levels of consensus—“proposed” and “approved.” Depending on the need for field evaluation or data collection, documents may also be made available for review at an intermediate consensus level.

Proposed  A consensus document undergoes the first stage of review by the healthcare community as a proposed standard or guideline. The document should receive a wide and thorough technical review, including an overall review of its scope, approach, and utility, and a line-by-line review of its technical and editorial content.

Approved  An approved standard or guideline has achieved consensus within the healthcare community. It should be reviewed to assess the utility of the final document, to ensure attainment of consensus (i.e., that comments on earlier versions have been satisfactorily addressed), and to identify the need for additional consensus documents.

Our standards and guidelines represent a consensus opinion on good practices and reflect the substantial agreement by materially affected, competent, and interested parties obtained by following CLSI’s established consensus procedures. Provisions in CLSI standards and guidelines may be more or less stringent than applicable regulations. Consequently, conformance to this voluntary consensus document does not relieve the user of responsibility for compliance with applicable regulations.

COMMENTS
The comments of users are essential to the consensus process. Anyone may submit a comment, and all comments are addressed, according to the consensus process, by the committee that wrote the document. All comments, including those that result in a change to the document when published at the next consensus level and those that do not result in a change, are responded to by the committee in an appendix to the document. Readers are strongly encouraged to comment in any form and at any time on any document.

Address comments to the Clinical and Laboratory Standards Institute, 940 West Valley Road, Suite 1400, Wayne, PA 19087, USA.

VOLUNTEER PARTICIPATION
Healthcare professionals in all specialties are urged to volunteer for participation in CLSI projects. Please contact us at exoffice@clsi.org or +610.688.0100 for additional information on committee participation.
Abstract

Clinical and Laboratory Standards Institute (CLSI) document AUTO8-P—Protocols to Validate Laboratory Information Systems; Proposed Guideline identifies important factors that laboratory managers should consider when developing a protocol for the validation of the Laboratory Information Systems (LIS). Also included are recommendations to help prepare validation protocols for assessing the accuracy and dependability of the LIS in storing, retrieving, and transmitting data.

The Clinical and Laboratory Standards Institute consensus process, which is the mechanism for moving a document through two or more levels of review by the healthcare community, is an ongoing process. Users should expect revised editions of any given document. Because rapid changes in technology may affect the procedures, methods, and protocols in a standard or guideline, users should replace outdated editions with the current editions of CLSI/NCCLS documents. Current editions are listed in the CLSI catalog, which is distributed to member organizations, and to nonmembers on request. If your organization is not a member and would like to become one, and to request a copy of the catalog, contact us at: Telephone: 610.688.0100; Fax: 610.688.0700; E-Mail: exoffice@clsi.org; Website: www.clsi.org
## Committee Membership

### Area Committee on Automation and Informatics

<table>
<thead>
<tr>
<th>Chairholder</th>
<th>Mayo Clinic</th>
<th>Salt Lake City, Utah</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paul J. Mountain, MSc, MT(ASCP)</td>
<td>Mary F. Burritt, PhD</td>
<td>Brian Richard Jackson, MD</td>
</tr>
<tr>
<td>Flamborough, Ontario, Canada</td>
<td>Rochester, Minnesota</td>
<td>ARUP Laboratories</td>
</tr>
<tr>
<td>David Chou, MD</td>
<td>Suzanne H. Butch, MA, MT(ASCP), SB</td>
<td>Paul W. Landesman, PhD</td>
</tr>
<tr>
<td>Vice-Chairholder</td>
<td>The University of Michigan</td>
<td>Abbott Laboratories</td>
</tr>
<tr>
<td>Univ. of Washington Medical Center</td>
<td>Ann Arbor, Michigan</td>
<td>Abbott Park, Illinois</td>
</tr>
<tr>
<td>Seattle, Washington</td>
<td>Al DeStefano</td>
<td></td>
</tr>
<tr>
<td>James V. Callaghan, MT(ASCP)</td>
<td>Sysmex Corporation</td>
<td>Michael D. McNeeley, MD</td>
</tr>
<tr>
<td>FDA Center for Devices and Radiological Health</td>
<td>Tucson, Arizona</td>
<td>MDS Metro Laboratory Services</td>
</tr>
<tr>
<td>Rockville, Maryland</td>
<td>Robert J. Dominici</td>
<td>Victoria, British Columbia, Canada</td>
</tr>
<tr>
<td>Randy R. Davis</td>
<td>Cholestech Corp.</td>
<td></td>
</tr>
<tr>
<td>Dade Behring Inc.</td>
<td>Alamo, California</td>
<td>Richard A. McPherson, MD</td>
</tr>
<tr>
<td>Newark, Delaware</td>
<td>Jefferson A. DuBois, PhD</td>
<td>Medical College of Virginia Hospital</td>
</tr>
<tr>
<td>Charles D. Hawker, PhD, MBA, FACB</td>
<td>NOVA Biomedical Corp.</td>
<td>Richmond, Virginia</td>
</tr>
<tr>
<td>ARUP Laboratories, Inc.</td>
<td>Waltham, Massachusetts</td>
<td></td>
</tr>
<tr>
<td>Salt Lake City, Utah</td>
<td>Louis J. Dunka, Jr., PhD</td>
<td></td>
</tr>
<tr>
<td>David A. Herold, MD, PhD</td>
<td>LifeScan, Inc.</td>
<td></td>
</tr>
<tr>
<td>VA (San Diego) Medical Center</td>
<td>Milpitas, California</td>
<td></td>
</tr>
<tr>
<td>San Diego, California</td>
<td>Robert H. Engel, PhD</td>
<td></td>
</tr>
<tr>
<td>Andrzej J. Knafel, PhD</td>
<td>Engel Associates</td>
<td></td>
</tr>
<tr>
<td>Roche Instrument Center AG</td>
<td>Duxbury, Massachusetts</td>
<td></td>
</tr>
<tr>
<td>Rotkreuz, Switzerland</td>
<td>Arden W. Forrey, Jr., PhD, FACB</td>
<td></td>
</tr>
<tr>
<td>Gary W. Kramer, PhD</td>
<td>University of Washington</td>
<td></td>
</tr>
<tr>
<td>National Institute of Standards and Technology</td>
<td>Seattle, Washington</td>
<td></td>
</tr>
<tr>
<td>Gaithersburg, Maryland</td>
<td>Masayoshi Hayashi</td>
<td></td>
</tr>
<tr>
<td>Rodney S. Markin, MD, PhD</td>
<td>Sysmex Corporation - Japan</td>
<td></td>
</tr>
<tr>
<td>Univ. of Nebraska Medical Center</td>
<td>Kobe, Japan</td>
<td></td>
</tr>
<tr>
<td>Omaha, Nebraska</td>
<td>Georg E. Hoffmann, MD</td>
<td></td>
</tr>
<tr>
<td>Advisors</td>
<td>Trillium GmbH</td>
<td></td>
</tr>
<tr>
<td>Michael G. Bissell, MD, PhD, MPH</td>
<td>Stephen Howlett</td>
<td></td>
</tr>
<tr>
<td>Ohio State University</td>
<td>Beckman Coulter, Inc.</td>
<td></td>
</tr>
<tr>
<td>Columbus, Ohio</td>
<td>Miami, Florida</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Terry Weakley</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Cerner Corporation</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Kansas City, Missouri</td>
<td></td>
</tr>
</tbody>
</table>
Subcommittee on Protocols to Validate Laboratory Information Systems

Sandy Pearson, MT(ASCP)
Chairholder
Center for Medicare & Medicaid Services
Dallas, Texas

Ulysses J. Balis, MD
Massachusetts General Hospital
Boston, Massachusetts

John Fuller
Werner Enterprises
Omaha, Nebraska

Brad Kowalski, MBA
Marshfield Laboratories
Marshfield, Wisconsin

Angela P. Locke, MT(ASCP)
Medical University of South Carolina
Charleston, South Carolina

Deborah Tillman
Misys Healthcare Systems
Johnstown, Pennsylvania

Quynh H. Vantu, MS, MT(ASCP) BB, CQA (ASQ)
Department of Veterans Affairs, Central Office
Washington, District of Columbia

Staff
Clinical and Laboratory Standards Institute
Wayne, Pennsylvania

David E. Sterry, MT(ASCP)
Staff Liaison

Donna M. Wilhelm
Editor

Melissa A. Lewis
Assistant Editor
## Contents

Abstract........................................................................................................................................... i

Committee Membership................................................................................................................... iii

Foreword........................................................................................................................................... vii

1 Scope........................................................................................................................................... 1

2 Introduction................................................................................................................................. 1

3 Definitions .................................................................................................................................. 2

4 Computer System Facilities ....................................................................................................... 4

4.1 Environmental Conditions and Safeguards for Proper System Operations ......................... 4

4.2 Preventative Maintenance for Computer Systems.................................................................... 6

4.3 Disaster Recovery..................................................................................................................... 8

5 Systems Programs ...................................................................................................................... 9

5.1 Adequate for Laboratory Services ......................................................................................... 9

5.2 Security/Access ....................................................................................................................... 9

5.3 Validation of New Programs or Changes to Existing Programs............................................ 16

6 Data............................................................................................................................................ 17

6.1 Registration and Requisition Information................................................................................ 17

6.2 Data Entry ................................................................................................................................ 19

6.3 Computer Calculations ........................................................................................................... 22

6.4 Autoverification of Transmitted Results................................................................................ 22

6.5 Generated Reports.................................................................................................................... 23

7 Quality Assurance Standards/CQI ............................................................................................ 26

7.1 User Responsibilities .............................................................................................................. 26

7.2 Database Maintenance ............................................................................................................ 33

7.3 Certification of Interfaces ....................................................................................................... 34

7.4 Problem Reporting and Tracking........................................................................................... 35

7.5 Train Staff ............................................................................................................................... 37

7.6 Postimplementation Monitoring ............................................................................................ 38

7.7 Documentation......................................................................................................................... 38

References..................................................................................................................................... 39

Additional References.................................................................................................................... 40

Appendix A. Software Design Recommendations for Manufacturers........................................... 41

Appendix B. Sample Change/Addition Request Form ................................................................... 51

Appendix C. Example of Change/Addition Request .................................................................... 52

Appendix D. Validation Script........................................................................................................ 53
Contents (Continued)

The Quality System Approach ........................................................................................................54

Related CLSI/NCCLS Publications .............................................................................................55
Foreword

The laboratory industry is quickly becoming computerized. There are many automated systems that a laboratory must interface with, both internal and external to the laboratory. Clinical and Laboratory Standards Institute (CLSI) has seven different approved standards that address individual portions of an automated laboratory system. These approved standards cover the path of workflow in a laboratory (preanalytical, analytical, postanalytical, and information management), yet there are no standards or guidelines that incorporate all these systems into a laboratory-wide validation process. This guideline contains recommendations for the preparation and execution of a Laboratory Information System (LIS) validation process. A Laboratory Information System (LIS) is also referred to as a Clinical Laboratory Information Management System (CLIMS) or Laboratory Information Management System (LIMS) in some current publications. For consistency, this document will use the term LIS throughout when referring to these types of systems.

Invitation for Participation in the Consensus Process

An important aspect of the development of this and all Clinical and Laboratory Standards Institute documents should be emphasized, and that is the consensus process. Within the context and operation of Clinical and Laboratory Standards Institute, the term “consensus” means more than agreement. In the context of document development, “consensus” is a process by which Clinical and Laboratory Standards Institute, its members, and interested parties: 1) have the opportunity to review and to comment on any Clinical and Laboratory Standards Institute publication; and 2) are assured that their comments will be given serious, competent consideration. Any Clinical and Laboratory Standards Institute document will evolve as will technology affecting laboratory or healthcare procedures, methods, and protocols; therefore, it is expected to undergo cycles of evaluation and modification.

The Area Committee on Automation and Informatics has attempted to engage the broadest possible worldwide representation in committee deliberations. Consequently, it is reasonable to expect that issues remain unresolved at the time of publication at the proposed level. The review and comment process is the mechanism for resolving such issues.

The Clinical and Laboratory Standards Institute voluntary consensus process is dependent upon the expertise of worldwide reviewers whose comments add value to the effort. At the end of a 90-day comment period, each subcommittee is obligated to review all comments and to respond in writing to all which are substantive. Where appropriate, modifications will be made to the document, and all comments along with the subcommittee’s responses will be included as an appendix to the document when it is published at the next consensus level.

Key Words

Audit trail, interface, network, system, validation, verification
Protocols to Validate Laboratory Information Systems; Proposed Guideline

1 Scope

The laboratory industry is quickly moving into the era of electronic reports, transmission of information via the Internet, etc., and there is a need to develop guidelines that can provide consistency in the industry. The purpose of this guideline is to address the validation of LIS systems and any interface to an external system (e.g., Electronic Health Record System [EHRS] formerly known as a Hospital Information System [HIS], Point-of-Care Device [POCD], reference laboratory, Data Repository, instrumentation, Laboratory Automation System [LAS], or financial system) to ensure that information is accurate and reliable during sample accessioning, transmittal of test results, and throughout the system’s intended use. This guideline addresses the validation process as it relates to:

- data entry;
- data analysis;
- data verification;
- data transmission;
- data storage; and
- data retrieval.

The primary focus of AUTO8-P is the software interface within the clinical laboratory environment. Therefore, the recommendations presented in AUTO8-P are not directly applicable to over-the-counter devices or software on instruments.

2 Introduction

A Laboratory Information System (LIS) manages data related to test requisitions, patient demographics, and specimens. An LIS can either interface with the laboratory analytical and process instruments as the data management center or serve for data collection, reporting, transmission, and archiving. An LIS can also interface with other information systems (e.g., Electronic Health Record System [EHRS]) for the transmission of test requisitions and final test results.

As stated previously, Clinical and Laboratory Standards Institute (CLSI) has seven different approved standards that address individual portions of an automated laboratory system (path of workflow):

**AUTO1**: Laboratory Automation: Specimen Container/Specimen Carrier provides standards for the design and manufacture of specimen containers and carriers used for collecting and processing liquid samples for clinical testing in laboratory automation systems.

**AUTO2**: Laboratory Automation: Bar Codes for Specimen Container Identification provides specifications for use of linear bar codes on specimen container tubes in the clinical laboratory and for use on laboratory automation systems.

**AUTO3**: Laboratory Automation: Communications with Automated Clinical Laboratory Systems, Instruments, Devices, and Information Systems facilitates accurate and timely electronic exchange of data and information among the automated laboratory elements.

**AUTO4**: Laboratory Automation: Systems Operational Requirements, Characteristics, and Information Elements defines system status information, which supports laboratory-automated systems.
AUTO5: *Laboratory Automation: Electromechanical Interfaces* defines a standard, compatible connection between instruments and automation technology that will enable the user to create an automated laboratory environment.

GP19: *Laboratory Instruments and Data Management Systems: Design of Software User Interfaces and End-User Software Systems Validation, Operation, and Monitoring* describes factors to be considered when developing new software-driven systems and selecting software user interfaces. Included are simple rules to help prepare validation protocols for assessing the functionality and dependability of software.

POCT1: *Point-of-Care Connectivity* provides a design framework for workstations and interfaces with an LIS.

<table>
<thead>
<tr>
<th>Path of Workflow</th>
</tr>
</thead>
<tbody>
<tr>
<td>Preanalytical</td>
</tr>
<tr>
<td>AUTO 1</td>
</tr>
<tr>
<td>AUTO 2</td>
</tr>
<tr>
<td>AUTO 3</td>
</tr>
<tr>
<td>AUTO 4</td>
</tr>
<tr>
<td>AUTO 5</td>
</tr>
</tbody>
</table>

AUTO8-P provides guidance for the development of a validation system for data management, which will incorporate all interfacing systems, both inside and outside of the laboratory. It identifies those elements that should be included in a validation system and the critical areas that should be considered in the validation process.

In the modern clinical laboratory, it is necessary for a laboratory to use and interface with different automated systems. It is important that laboratory staff validate the integration/operation of all automated systems to ensure the accuracy of all test information.

AUTO8-P specifications are also intended to complement the interrelated CLSI/NCCLS standards developed by other automation subcommittees and to support overall operational goals for future development in laboratory instrumentation and automation.

### 3 Definitions

**Accuracy** – Closeness of agreement between a test result and the accepted reference value (ISO 3534-1)\(^1\);

**Audit trail** – 1) Data in the form of a logical path linking a sequence of events, used to trace the transactions that have affected the contents of a record; 2) A chronological record of system activities that is sufficient to enable the reconstruction, reviews, and examination of the sequence of environments and activities surrounding or leading to each event in the path of a transaction from its inception to output of final results.\(^4\)

**Completeness** – The property that all necessary parts of the entity are included; **NOTE**: Completeness of a product is often used to express the fact that all requirements have been met by the product.\(^5\)

**Computer system security** – The protection of computer hardware and software from accidental or malicious access, use, modification, destruction, or disclosure. Security also pertains to personnel, data, communications, and the physical protection of computer installations.\(^2\)
Consistency – The degree of uniformity, standardization, and freedom from contradiction among the documents or parts of a system or component.\textsuperscript{2}

Correctness – 1) The degree to which software is free from faults in its specification, design, and coding; 2) The degree to which software, documentation and other items meet specified requirements; 3) The degree to which software, documentation, and other items meet user needs and expectations, whether specified or not.\textsuperscript{2}

Efficiency – The degree to which a system or component performs its designated functions with minimum consumption of resources.

Expandability – The ease with which a system or component can be modified to increase its storage or functional capacity.

Expert system – A software system consisting of a knowledge base, inference engine, and explanation unit; \textit{NOTE:} The knowledge base contains rules from experts, the inference engine uses this knowledge to reach conclusions given certain facts, and the explanation unit serves to explain how the conclusions were reached.

Flexibility – The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed.

Interface – 1) A shared boundary between two functional units, defined by functional characteristics, common physical interconnection characteristics, signal characteristics, and other characteristics, as appropriate. The concept involves the specification of the connection of two devices having different functions; 2) A point of communication between two or more processes, persons, or other physical entities; 3) A peripheral device which permits two or more devices to communicate.\textsuperscript{4}

Interoperability – The ability of two or more systems or components to exchange information and to use the information that has been exchanged.

Maintainability – The ease with which a software system or component can be modified to correct faults, improve performance or other attributes, or adapt to a changed environment.\textsuperscript{2}

Network – 1) An arrangement of nodes and interconnecting branches; 2) A system (transmission channels and supporting hardware and software) that connects several remotely located computers via telecommunications.\textsuperscript{4}

Operating system – Software that controls the execution of programs, and that provides services such as resource allocation, scheduling, input/output control, and data management; \textit{NOTE:} Usually, operating systems are predominantly software, but partial or complete hardware implementations are possible.\textsuperscript{4}

Petri net – An abstract, formal model of information flow, showing static and dynamic properties of a system; \textit{NOTE:} A Petri net is usually represented as a graph having two types of nodes connected by arcs, and markings indicating dynamic properties.

Portability – The ease with which a system or component can be transferred from one hardware or software environment to another.

Quality assurance – 1) The planned systematic activities necessary to ensure that a component, module, or system conforms to established technical requirements; 2) All actions that are taken to ensure that a development organization delivers products that meet performance requirements and adhere to standards and procedures; 3) The policy, procedures, and systematic actions established in an enterprise for the...
purpose of providing and maintaining some degree of confidence in data integrity and accuracy throughout the life cycle of the data, which includes input, update, manipulation, and output; 4) The actions, planned and performed, to provide confidence that all systems and components that influence the quality of the product are working as expected individually and collectively.

Reliability – The ability of a system or component to perform its required functions under stated conditions for a specified period of time.

Requirement – 1) A condition or capability needed by a user to solve a problem or achieve an objective; 2) A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents; 3) A documented representation of a condition or capability as in 1) or 2).

Reusability – The degree to which a software module or other work product can be used in more than one computer program or software system.

Specification – A document that specifies, in a complete, precise, verifiable manner, the requirements, design, behavior, or other characteristics of a system or component, and often, the procedures for determining whether these provisions have been satisfied; contrast with requirement.

System – 1) People, machines, and methods organized to accomplish a set of specific functions; 2) A composite, at any level of complexity, of personnel, procedures, materials, tools, equipment, facilities, and software. The elements of this composite entity are used together in the intended operational or support environment to perform a given task or achieve a specific purpose, support, or mission requirement.

Testability – 1) The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met; 2) The degree to which a requirement is stated in terms that permits establishment of test criteria and performance of tests to determine whether those criteria have been met.

Usability – The ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.

Validation – Confirmation, through the provision of objective evidence, that requirements for a specific intended use or application have been fulfilled (ISO 9000); NOTES: a) WHO defines validation as “the action of proving that a procedure, process, system, equipment, or method used works as expected and achieves the intended result.” (WHO-BS/95.1793); b) Establishing documented evidence which provides a high degree of assurance that a specific process will consistently produce a product meeting its predetermined specifications and quality attributes.

Verification – Confirmation, through the provision of objective evidence, that specified requirements have been fulfilled (ISO 9000).

4 Computer System Facilities

4.1 Environmental Conditions and Safeguards for Proper System Operations

Laboratory information systems have shared in the dramatic increase in computing power over the past two decades. There has been a corresponding relaxation in the stringency of regulating the environmental conditions of the computing facility. Where early on it might have resembled a “clean room,” the accepted environment now is that of “the office.” Similarly, considerations for fire safety and assurance of a steady power supply resemble those for an office, the level of protection dictated primarily by the
value of the data and the need for uninterrupted service rather than physical requirements of large mainframes.

### 4.1.1 Ventilation, Humidity, and Ambient Temperature

Vendors of laboratory information systems are essentially software providers: they refer questions about environmental conditions for computing to the hardware manufacturers. Specifications for your computer systems/servers should be obtained from the manufacturers, but representative parameters can be used for illustrative purposes. Typical “office environment” parameters include:

- **Humidity:** 20 to 80% (noncondensing)
- **Temperature:** 18 to 28 °C
- **Ventilation:** Sufficient to maintain desired humidity and temperature. The allowable rate of change should also be specified.

These are very conservative settings. Damage to magnetic media can begin at 37.8 °C, and unrecoverable damage begins approximately at 48.9 °C. The Policy and Procedure manual for the LIS should specify the acceptable limits, the frequency of monitoring, and the orderly response in the event of loss of control of one of these parameters.

### 4.1.2 Fire Prevention

With respect to data integrity, the most important single measure is regular data back-up. The data archive should NOT be kept in the computing area. Ideally, the back-up system and archive should be off-site or at least separated from the computer area by fire-resistant construction, sufficient to allow at least one-hour protection.

Consideration should be given to construction of both the computer system and the structure in which it is housed, as well as the connections between the facility and the outside, including but not limited to doors, firewalls, ventilation system, and electrical/communication cabling. Detailed recommendations with respect to these aspects of fire prevention are available from professional fire prevention/protection organizations (e.g., *Standard NFPA 75, Standard for the Protection of Electronic Computer/Data Processing Equipment* from the National Fire Protection Association). The final configuration of safety measures must meet or exceed the most stringent requirements of any controlling authorities (hospital, municipality, or other).

There are several aspects, which should be explicitly evaluated:

- **Automatic detection system**
  - Positioning: floor, ceiling, vents, other
  - Automatic shutdown of electrical equipment
  - Recipient(s) of alert
  - Action plan if alert given

- **Extinguishing systems**
  - Portable Class C fire extinguisher for limited fires
  - Automatic suppression system
  - Automatic sprinkler system
  - Potential for water damage
- Gaseous Total Flooding Extinguishing Systems
  Some agents can leave a chemical residue.
  Some agents, in large volume, are toxic.

- Ventilation

  Smoke from fires elsewhere in the building can damage computing equipment. The expense of modifying the ventilation system must be balanced against the risk and potential damage.

- Action plan for facility personnel

  In addition to the general fire response policy, the policy should address powering down (“pulling the plug”) either in an orderly fashion or abruptly, depending on the urgency of the situation.

4.1.3 Power

Given the importance of laboratory information systems, **BOTH** an uninterruptible power supply (UPS) and a line conditioner (possibly in one unit) are essential. Consult both the computer manufacturer and the UPS manufacturer to make sure the power/voltage/current characteristics of the computer, the UPS, and the electrical supply are matched. A UPS can provide minutes to hours of back-up time, a function mainly of price. The LIS group should determine the minimum amount of back-up time required to respond (it will take longer at night) and safely power down the system. Consult your institution’s engineers about the availability of fail-safe electrical lines for emergency power: even if the institution has a capable back-up generator, it is still advisable to have a local UPS system for the computer.

The Policy and Procedure Manual for the LIS should have an action plan including:

—How to monitor the UPS:
  
  - Many perform self-checks, but the indicator panel should be checked on a regular basis. Some can be connected to a computer, and interrogated in more detail.
  
  - Track battery life (consult the manufacturer) and replace before an indicator warning. Batteries are typically NOT readily available and might require expert installation, so the policy should provide forward planning.

—Sign(s) of power failure
—Verification that the UPS (and/or fail-safe line) is working
—Whom to notify
—Who determines that the computer system should be powered down?
—The procedure for an orderly power-down (presumably already in the LIS procedure manual).

Wires/computer cables must not run along the traffic paths on the floor or across accessible areas of crowded desktops. Accidental tension on a cord, especially on a floor, can generate not just an unavoidable power-down, but also a serious injury.

4.2 Preventative Maintenance for Computer Systems

The requirement of keeping all computer systems being used within the laboratory environment in adequate operating mode is critical to the continued smooth operation of the laboratory. With the diverse nature of laboratories and computer systems, this area of preventative maintenance should include (but not be limited to) the following items:
• Computer Hardware – All hardware vendor recommendations as to the proper and timely maintenance of said hardware must be followed. A method of documenting this maintenance should be created to help ensure compliance with those recommendations. Items included in the maintenance log should be: hardware identifier, person(s) involved, date(s)/time(s), what’s being done, and any miscellaneous information. The types of hardware included should be: mainframe system(s), personal computer(s), printer(s), bar-code equipment, communication/networking equipment, UPS Systems, and line filter equipment. It must be noted that some equipment manufacturers are beginning to recommend that no period maintenance is required for their equipment. In this event, a single line item in the maintenance log stating this is sufficient.

• Computer Software – This area is generally not considered preventative maintenance, however, it is critical that laboratories stay current with software (both operating systems and user software) versions. Continuing to operate on older versions of software may compromise a vendor’s ability to adequately support the laboratory.

4.2.1 Operating Limits Documented

All computer systems contain various limits that will restrict the operational efficiency of that system. The laboratory must be aware and document what these potential limits could be to help ensure the continued smooth operation of the various computer systems. These limits could include:

• Number of licensed users for the operating system. Generally, this is a hard limit that cannot be exceeded.

• Number of licensed users for the laboratory software. Depending upon the vendor, this may or may not be a hard limit. In the event of a soft limit, the laboratory may actually have more users on the system than are permitted by the license.

• Available disk space. On many computer systems, this can become a critical bottleneck and slowdown as the file sizes grow beyond a certain limit (e.g., most Unix-based operating systems become slower after more than 90% of the available disk space has been used).

• Back-up capacity. Many computer systems have disk drives added to them over time with no regard as to the time it takes to back them up to removable media. Additional back-up capacity may be required.

4.2.2 Downtime of System/Emergency Back-up Plan

All organizations should create and test comprehensive plans concerning how the laboratory operations will continue to function in the event of a computer being down—scheduled or unscheduled. Laboratory personnel should be familiar with these plans and able to implement them in a rapid timeframe if needed.

These plans should include the following variations:

• scheduled, prime-time down, mainframe (e.g., Tuesday, 1:00 p.m.);
• scheduled, off-time down, mainframe (e.g., Sunday, 1:00 a.m.);
• unscheduled, prime-time down, mainframe (e.g., Tuesday, 1:00 p.m.);
• unscheduled, off-time down, mainframe (e.g., Sunday, 1:00 a.m.);
• scheduled, prime-time down, connectivity (e.g., Tuesday, 1:00 p.m.);
• scheduled, off-time down, connectivity (e.g., Sunday, 1:00 a.m.);
• unscheduled, prime-time down, connectivity (e.g., Tuesday, 1:00 p.m.);
• unscheduled, off-time down, connectivity (e.g., Sunday, 1:00 a.m.); and
• extended hours downtime (e.g., 24 hours).

The various plans are necessary because all laboratories have different levels of staffing (both laboratorians and information systems personnel) throughout the day.

4.2.3 Communication With Clients

In today’s laboratory environment, many sites are electronically connected to both reference laboratories and clients. It is becoming critical that plans are in place to help ensure connectivity, reliability, and smooth operation of these connections. A comprehensive plan should be created to test all connections on an annual basis. This plan should include:

• physical connection;
• verify patient demographics are accurate;
• verify patient orders are accurate;
• verify patient results are accurate; and
• verify patient billing is accurate.

A plan should also be created to help manually communicate all planned downtimes to the various external connections, and what to do in the event of unplanned downtime.

4.2.4 Client/Server Networks

Similar to communication with clients, most laboratory computer systems are based on client/server architecture that is critical to ensure smooth operations. All client/server systems have some very hard limits as to the number of clients per segment, the bandwidth available for connectivity, and the distance from wiring endpoints. A laboratory must be aware of these restrictions and plan for growth. Concurrently, the technology being developed for network connectivity is rapidly changing and advancing. A laboratory must stay involved with changes in technology and plan for periodic upgrades.

4.3 Disaster Recovery

A comprehensive plan must be created for the laboratory’s “worst case computer scenario.” For most laboratories, this would be the destruction of the computer facility only (the patient seeing facility and the laboratory facility are still operational) that would cause computer disruption of greater than 24 hours. The area of disaster recovery should be divided into two distinct areas: required daily operations to ensure minimal disruption of computer services and the contingency plan to be performed in the event of a disaster. The Daily Operational plan should include the following items:

• system back-up plans of software and data;
• physical location, security, and accessibility of the back-up media (e.g., off-site storage, elements protection, restricted access);
• the retention criteria of the back-up media (i.e., how long the back-up media are kept, conversion plans in the event of back-up methodology changes, and verification of long-term storage); and
• verification plans to ensure that the back-up media are readable and to identify any loss of data.
The Disaster Recovery plan should include:

- contact personnel (listing who is responsible for what);
- obtaining a site for housing the replacement equipment (including network/communications connectivity);
- obtaining replacement equipment;
- restoration of software systems;
- restoration of databases;
- identifying and reconstructing missing data;
- verification of system for operational use; and
- a process and periodic timetable to verify that this plan works.

All Daily Operational and Disaster Recovery plans should be reviewed on an annual basis to ensure accuracy (reflects the current computer environment), effectiveness (does the plan actually work), and conformity to the various regulatory agencies under which the laboratory is licensed (JCAHO, CAP, FDA, etc.).

**NOTE:** Due to the rapid changes in technology over the past few years, the subcommittee is unable to provide recommendations regarding the copying of stored data to newer media. In the past, magnetic tape degraded over time requiring replacement (the time factor changes depending upon storage conditions, such as humidity and temperature control). The newer CD-ROM media do not appear to degrade unless the media are not stored in a controlled environment. The replacement of these media is most likely going to depend upon the hardware readers involved and will be up to the particular hardware vendors involved.

The validation of the databases, after moving them back onto an accessible environment, is strictly dependent on what the software/hardware vendors are capable of doing. At a minimum, the ability to restore the databases on the active/back-up system and the ability to conduct the normal software functions (e.g., inquiry, result entry) would be required. After the databases have been successfully validated, they can then be implemented (copied) to the production environment. This method helps ensure that no file corruption has been introduced by the restoration process.

5 Systems Programs

5.1 Adequate for Laboratory Services

5.1.1 Meets Accreditation/Regulatory Requirements

Each laboratory is responsible for knowing and complying with all applicable rules governing privacy and confidentiality in its jurisdiction.

5.2 Security/Access

The security of laboratory data and the knowledge of who has access to these data are paramount in keeping the data safe and confidential. The number of portals or entry points that can access the data
should be minimized. The more methods of access that exist, the easier it is for unauthorized personnel to view these data.

5.2.1 Physical Security Access to CPUs and Disk Storage Device Locations

Physical access to the facility that houses the CPU and disk storage devices should be as secure as the data that reside on this equipment. The following measures are suggestions as to how to ensure that access to this equipment is controlled:

- Locked room with keys distributed to personnel who have signed a document accepting responsibility for the key.
  - Procedures should be in place prohibiting access to the room when an employee has changed job positions, has been terminated, or has left on unfriendly terms.
    - Locks are immediately changed.
    - Personnel are supplied with new keys.
- Room locked with a keypad combination access. The combination is given to personnel who require access to this room.
  - Periodically, the combination should be modified.
  - Notification of change should be done via voicemail rather than e-mail.
  - The combination should be changed immediately upon employee change in job positions, termination of employee, or if the employee left on unfriendly terms.
- Room that is secured with an employee bar-coded badge scanner. When an employee is hired for a position that requires access to this hardware, his/her employee badge bar code is activated providing access to this door scanner.
  - Employee badge bar code should be inactivated immediately upon employee change in job positions, termination of employee, or if the employee left on unfriendly terms.

5.2.2 Protected Access Codes

Access codes should be in place to ensure that only authorized personnel are reviewing, entering, or modifying the data. Even with access codes in place, all data seen by personnel should be noted, preferably by an automated audit trail that is not accessible to anyone but the Information Technology Staff.

Access codes refer to the user’s entry of a set of characters that identifies the user as a valid participant on the LIS. Typically, information systems will contain username and password as their main access to the system. Only the assigned user should have knowledge of these codes. It is recommended that generic codes not be utilized when accessing the LIS. Access codes are meant to be secure and are not distributed to other personnel either in the laboratory or throughout the entire facility. Periodic modification of the codes should be performed to ensure security of user logins.

In order to validate that these access codes are protected, it is recommended that the following steps be taken and documented:

- Verify that the current username and/or password are valid.
- Verify that one user cannot access the system by using his/her password with another’s username.
• Verify that at a minimum, the password does not display on the screen during data entry. Typically, the username does appear on screen as you are entering the characters.

• The user should make his/her best effort to ensure that all security guidelines as outlined by CLIA, CAP, and HIPAA or applicable regulatory agency, including federal, state, and local policies, are being followed.

• If an invalid password is entered into the system, a warning message stating that the user has attempted to enter an invalid access code should appear. Validate that this message displays. Some LIS vendors will disallow the user from entering the system if they continue to try to enter invalid access codes. If your vendor does provide this feature, validate that this parameter is working according to the setting defined in the system. Also, if there is an audit trail of these attempts, verify that these are being logged.

The more methods of access to the system, the easier it is for unauthorized personnel to gain access to the data. Open ports and active modems pose two of the easiest ways to access data without anyone’s knowledge. Measures should be in place to allow the following to occur when personnel outside of your facility (your LIS vendor) are requesting permission to your system:

• Only needed ports should be left open and modems should be turned off when not in use by authorized personnel.

• Monitoring software should be in place on all machines that could access data. Monitoring software should be able to detect and log access on all ports that need to be left open.

• Monitoring open ports can be as simple as checking the system logs to determine what has been accessing the port. However, there are specialized software packages available that will track all incoming and outgoing packets on the open ports.

• Any unusual activity should be noted and investigated.

• Work with the vendor to determine what ports must remain open in order to properly utilize the system.

Policies need to be part of the Standard Operating Procedures (SOP) that outline the procedure to follow when the LIS vendor requires access to the system. Those policies may include the following:

• Ensure that the LIS vendor understands the dial-in procedure so that there is no delay in them obtaining access to the system during critical downtimes.

• Below are examples of procedures that may be posted on the system for display when a vendor has dialed in.

  1. Access requires a passcode. Call xxx-xxx-xxxx to obtain a passcode. Passcode expires one minute after it is issued. The user will get a confirmation of passcode accepted if he/she has typed the passcode correctly.

  2. THE USER MUST CONTACT THE SITE BEFORE LOGGING IN TO BE GIVEN ACCESS TO THE MODEM. DURING THE DAYTIME, CONTACT XXX-XXXX FOR PRIVILEGED (LOGIN) ACCOUNT ACCESS. OFF HOURS, CALL XXX-XXXX.

   For every three failed login attempts, your username will be locked out the system for 15 minutes.
Steps to Login:

a) Enter username:xxxxx and password:xxxxx.

b) At the second username prompt, enter:xxxx password:xxxx.

c) Then enter your name, service request number (problem you are working on), and summary of purpose of login.

Use of the data by LIS vendors for troubleshooting purposes should be kept to a minimum. Should a vendor need a copy of some or all data in order to duplicate situations, the data should be sent in a sealed container to be certain that only the vendor retrieves the data. Upon completion of the task, the vendor should return the data in a sealed container and destroy all copies of the data. Vendors may be required to retain records based on FDA and ISO regulations. LIS vendors will have procedures in place to document receipt and return of data. Also, the LIS vendor will document that copies of media have been destroyed after the media are returned. Do not hesitate in requesting this procedure.

- The site should document when the media are shipped and also when the media are returned. Information concerning the following items should be retained:
  - date sent and authorizing person;
  - date returned and receiver;
  - type of media sent;
  - reason/purpose for sending media;
  - vendor contact person – who will be receiving the media;
  - method of delivery, tracking number, copy of shipping invoice;
  - summary of information residing on media; and
  - disposition of media upon return.

The facility should have a procedure covering the scenario when an employee is no longer employed by the laboratory department. Notification should also be given to departments outside of the laboratory such as information services, facilities, and administration, who may control some of the software access methods to the laboratory system.

- Usernames, passwords, tech codes, or any identifying codes are inactivated from the software. These codes should not be deleted, but merely inactivated. This is to ensure that the users can no longer access the system, but authorized personnel can still audit any tasks that the former user had performed in the system.

- If the site has provided these employees with access to the physical location of the hardware, then measures should be initiated in order to inactivate any access to the facility. See Section 5.2.1.

Facilities must reference and comply with the Health Insurance Portability and Accountability Act (HIPAA) regulations 45 CFR Parts 160, 162, 164 – Health Insurance Reform Security. More information can be found at the following websites:

http://www.hhs.gov/ocr/hipaa/
http://aspe.os.dhhs.gov/admnsimp/index.shtml

A checklist can be found in Appendix A to Subpart C of Part 164 – Security Standards: Matrix of the Health Insurance Portability and Accountability Act (HIPAA) that can guide the system manager through
the appropriate documentation, procedures, and processes that should be maintained for the security of the data.

5.2.3 Authorized Personnel Access to Data

There are many steps in an LIS system to assign authorized personnel access to the data.

**Review job requirements of personnel and/or groups (i.e., clerical vs. technical).**
The first step in assigning access to the system is to determine the various operations of the personnel in the laboratory as well as those outside the laboratory. Prepare a listing of each operation group such as clerical, data entry, and general result entry. This may need to be further categorized to specific departments (i.e., Microbiology and Blood Bank). List each group’s specific needs as they relate to their job requirements. For any users outside of the laboratory environment—infectious disease or nursing—a review of their access needs to the laboratory system for patient reports or internal reports should be included in the evaluation.

**Document all available options in the system.**
Review a full listing of all available LIS functions and/or options in order to determine which will be used at the facility and how they will be assigned to personnel.

**Determine the needs of personnel and/or groups.**
For each group or department, determine the specific level of access needed. For example:

- Will the group need to access patient data, and if so, should they be accessing all patient data or will they be restricted to certain data?

- Will the technologists entering results be restricted to entering all types of results (i.e., General Laboratory [Chemistry, Hematology], Microbiology, and Blood Bank), or will they be restricted to a particular section?

**Assign security levels to various operations to be used.**
When a complete review of the available functions has been accomplished, develop a logical plan as to how security levels will be assigned to each person who is allowed access to the system.

**Determine password assignment.**
Establish within the institution a password policy. This policy should include examples of acceptable and unacceptable passwords. State within that policy the following:

- Length of password – recommend four to eight characters

- Password content
  - All alpha
  - All lower case
  - All upper case
  - Mixture of lower and upper case
  - Ability to use special characters

- Data range when password is changed

- Can the password be reused

Some institutions may opt to use a “strong” password concept.
Example: Password must be a minimum of eight characters, two of which need to be lower case, and two of which need to be special characters.

Users should avoid using birthdates, anniversary dates, family member names, or any other familiar terms that could be associated with the user.

Also include with this policy a logical scheme for defining each access code into the system.

- Will the users have a username/password?
- Is the username based on some combination of last name and department or location?
- Should the username follow the same format as other logins (i.e., network login)?

**Associate the username/password to functions.**

Because of the sensitive nature of the information stored and processed on the LIS system, it is important to have restrictions or levels of security established for all applications or functions within the system. Each individual’s security level is based on his/her position, job description, and type of patient information access required to perform his/her job. This security access may not just be limited to the LIS systems, but also other related applications such as patient administrative and financial systems. Based on the institution’s configuration, the user may need to log in to a network prior to accessing the LIS system. The same security guidelines followed for the LIS should also be applied to these other applications and network logins.

When all LIS functions have been reviewed and a login scheme has been developed, the next step is to associate the specific users to the appropriate operations. It is recommended that a record of each person be kept on file to track what level of access was provided to the user. Passwords should not be recorded. However, the username can be associated with each person for easy retrieval of this information.

**Test various levels.**

A plan will also need to be created to document that the assigned security levels are working as expected. The system manager could create a generic login to test the various operations assigned based on the security level. This generic login could be reused to test the different levels of security that will be assigned to various users.

**Provide documentation for user logins.**

Record the date each user is given his/her specific login. The system manager will need to determine what type of signature, manual or electronic, is required to be kept permanently. The system manager may also need to document that the assigned username/password works as expected based on the user’s feedback.

**Verify password changes, if needed.**

The system manager will assign the initial username/password in order for the user to gain access to the system. Establish a mechanism in which the user must immediately change the password to one that is only known to that individual. The system manager may need to verify that the password has been changed. To verify this, the site may require a signature to document that the password has been changed. The system manager may then verify by attempting to login as the user with his/her original password to confirm that the password has been changed.

It is recommended that a policy stating that passwords expire is in place, thus requiring the users to define a new password periodically.
5.2.4 Maintenance of Audit Trail to Track Personnel Usage

When the security levels and access to the system have been created, tested, and verified, the system manager will need to monitor access to the system as well as changes to the database. If the system has the capability of creating files to capture such data, the system manager should test and verify that the files created are correct.

HIPAA (Health Insurance Portability and Accountability Act) requires healthcare organizations to establish formal, stringent procedures on how health information is handled internally and by business associates. It requires the retraining of all employees who handle identifiable information in new procedures governing the use and disclosure of information. It requires physical safeguards to protect computer systems and information technology to control and monitor access to data and secure data in transit. The main goals of the act are to:

- Curtail fraud and abuse in health care.
- Improve health insurance portability by building on the Consolidated Omnibus Budget Reconciliation Act (COBRA).
- Mandate administrative simplification through the use of standard code sets, unique identifiers, and standard EDI transaction sets.
- Prescribe privacy and security standards to safeguard the privacy of personal healthcare information.

To comply with HIPAA regulations, it is recommended that the system logs:

1. Access to the system (security log);
2. Access to patient data (patient file); and
3. Changes to the database (maintenance file).

Security Log
The ‘security’ log captures data at the time a user gains access to the system and when the user exits the system. This file also logs when users modify their passwords. Once these logs have been created, the system manager should test and verify that the data were captured appropriately. This would include any login access to the system, changing of passwords, and attempts to access the system with invalid codes.

It is recommended that the system manager review the security log on a daily basis to monitor data access failures. From this consistent monitor, the system manager can determine patterns of unauthorized personnel attempting to access the system without an appropriate username and/or password. If there is a consistent username and password failure, the system manager will disable or modify the login access.

Patient File
The ‘patient’ file captures incidence of access to patient data including any type of inquiry of data and printing or faxing of patient results. The system manager should test each operation that allows an inquiry of patient data to confirm that access is logged. A review of all patient reports that can be generated manually or automatically should be tested to confirm that report accessibility is being logged.

It is recommended that the system manager monitor this log on a weekly basis. From this, the system manager can determine if personnel are intentionally inquiring about a specific patient’s results outside of normal laboratory workflow.
Maintenance File
The ‘maintenance’ file includes modifications to any portion of the database. It is recommended that each database option be reviewed and tested to confirm that the maintenance file captures the changes. The system manager will need to verify that the appropriate data are captured in each field for the various logs.

It is recommended that the system manager monitor this log on a weekly basis. If multiple users are allowed access to maintenance and/or dictionary files, it is important for the system manager to understand that changes are being made to the system, why those changes are being made, and also who is performing those changes.

The final step in monitoring the files is to determine long-term storage of the data. The facility should establish procedures to comply with various regulatory agencies that govern the length of time and site (facility) for storing such data.

5.2.5 Virus Protection
Computer systems should be monitored and protected against malicious intrusions and attacks. Techniques for this monitoring and protection include the use of firewalls, virus protection software, removing unwanted network functions, and appropriate network filters and monitors such as TCP wrappers.

It is recommended that virus software be installed on all devices that could access laboratory data. Where regulatory requirements permit, virus software should be updated on a daily basis and be running at all times.

Validate that the virus software runs daily on both the end-user devices and servers. Where regulatory requirements permit, ensure that updated virus software is loaded onto each PC on a daily basis. If virus software updates are not automatic, ensure that there are policies in place to install, run, and document that virus-checking software has run on a daily basis.

5.2.6 Data Encryption Technologies
Any time data that contain identifiable patient information are transferred outside of the institution, they should be encrypted. Encryption refers to scrambling data in a manner that only the sender and receiver of the data are able to reassemble the data into a usable format. Transmission to off-site locations by way of fax or other medium should be done only when authorized personnel are available in the remote location to handle these data.

5.3 Validation of New Programs or Changes to Existing Programs
5.3.1 Documentation
Periodically, the vendor will apply software updates to the system. Vendors may forward the following with each software update:

- user documentation;
- program documentation;
- recommended client test plans; and
- prerequisites for loading of software (additional hardware and software).

Review all literature submitted by the vendor on the new release. As part of the process to migrate this new software, ensure that the laboratory:
• performs general, functional, workflow testing for all departments in the laboratory.

• establishes testing guidelines according to the Standard Operating Procedures. The laboratory may use the General Testing Guidelines provided by the vendor as a template.

• tests all new features applicable to the laboratory.

• reviews all Release Notes and/or Package Listings provided by the LIS vendor to identify any software updates which may have a major impact on laboratory operations or which may help improve workflow.

### 5.3.2 Communication to All Users

Occasionally, vendors will perform troubleshooting on the system. Typically, they may perform a fix on the system. Develop a policy and review with the vendor outlining communication processes when a vendor needs access to your system. It is recommended that this policy describe the steps that the vendor will follow to gain access to the system and the manner in which they will communicate the outcome of this investigation. See Section 5.2.2 for more details.

Prior to migrating any software changes (major or minor releases) to the production area, a review of all the new functionality should be performed by laboratory personnel. The outcome of this review will establish the new feature(s) to be utilized by the facility. The site performs testing of a new feature(s) as it pertains to the facility’s Standard Operating Procedures.

If the vendor provides testing guidelines, it is recommended that these be used only as a basis for the testing process. The final testing plans should represent the Standard Operating Procedures and not the testing recommendations of the vendor.

During this testing, establish training guidelines outlining the curriculum for educating existing staff on new features. Prior to migrating the new features to the production area, it is recommended that all staff be trained on new features. Records should be kept on each employee that certify proper training was performed on each new feature. These training sessions and materials should be documented. These materials can then be used for training of new employees.

Coordinate with the vendor as to when this software will be migrated into production. Typically, the vendor will require written notification that all of the recommended testing has been performed, and that the system manager wishes to move the code to production. Notify all users as to the date of time of migration and any downtime associated with this migration.

### 6 Data

Data entered into the system should be validated. Each function used to enter data should be validated. The LIS user guides and support documents provided by the vendor are a vital resource. These documents should be used with site-specific guidelines to validate that the LIS is performing as expected. (Examples of Change/Addition Request Forms can be found in Appendixes B and C.)

#### 6.1 Registration and Requisition Information

Whether specimen registration occurs using the LIS or an external Electronic Health Record System (EHRS), formerly known as an HIS, interfaced with the LIS, the accuracy of information associated with specimens is important. Accurate registration and requisition data ensure correct specimen identification and processing (see the most current version of CLSI/NCCLS document LIS4—Standard Guide for Documentation of Clinical Laboratory Computer Systems). Each laboratory should identify the data
fields, specific to their LIS, to be validated. If data fields accept more than one specific data value, then a representative sample or adequate percentage of acceptable entries should be tested.

If registration is performed using an interfaced EHRS system, then only the information transmitted to the LIS should be validated. Print screens of the EHRS registration information can be used to validate the data in the LIS.

If specimen registration only occurs in the LIS system, then compare the electronic LIS data with the original paper documentation used for the data entry.

The following is a list of data fields that should be validated:

- name or other alpha identifier;
- unique specimen identifier or medical record number;
- age or date of birth;
- sex;
- date of service;
- financial number, if applicable;
- ordering physician;
- insurance information, if applicable.

Other information that may be validated includes:

- diagnosis or currently recognized version of ICD9 code;
- comments or notes;
- specimen source;
- user ID that performed data entry;
- Social Security Number; and
- address.

Requisition information is data specific to the tests to be performed on the specimen. These data may be entered into the LIS electronically, via an external interfaced order entry system, or manually entered into the LIS using a written order or standard order form. Each data field required at the time of order entry should be validated for accuracy. If orders are received from an external system, then the print screen from this system may be used to validate the data in the LIS. The paper requisition can be used if orders were entered directly into the LIS. It is recommended that all test procedures be validated at the time of system installation and then periodically thereafter.

The following is a list of data fields that should be validated, as applicable to the specimen or test ordered:

- name or other alpha identifier;
- unique specimen identifier or medical record number;
- age;
- location;
- collection date and time;
- order status (future collection, in progress, in laboratory);
- unique system specimen identifier (accession, requisition, or bar-code number);
- ID of phlebotomist;
- currently recognized version of ICD9 code, if required;
- ordering physician;
- comments or special instructions;
prompts for additional information (volume, collection hours, weight);
• test(s) ordered;
• collection labels; and
• different order priorities (stat, routine, timed).

The following are some questions that may be asked to further validate the requisition information:

• Did the collection requirements print on the collection report and/or specimen label(s)?

• Did the comments or special handling instructions print on the collection report and/or specimen labels?

• Was the correct number of specimen labels generated by the system?

• Did the system process the order entry/specimen receipt step in a timely manner as compared to established time limits?\[12\]

• Is the patient consent information displayed in the appropriate data field?

• Is the callback information displayed correctly or printed on the appropriate report correctly?

• Did the test(s) load to the correct instrument or benchwork queue? Is the specimen information correct on the corresponding loadlist or worksheet?

• Did the system accept, if applicable, both the numeric and alpha-mnemonic code for the test to be ordered?

The system should be tested for any error messages or other system-defined parameters. The following are some examples:

• Verify that test(s) cannot be ordered based on patient age or sex.
• Verify that test(s) cannot be ordered for a specific status (stat) or on a certain shift.
• Verify that certain test(s) can only be ordered by a supervisor or by using a password.
• Verify that data fields with limited acceptable results will not accept any other entry.

Validation should include any special functions or features used in specimen processing and workflow (see the most current version of CLSI/NCCLS document LIS4—Standard Guide for Documentation of Clinical Laboratory Computer Systems). Special functions can include help, search, calendars, date/time stamps, free text speed typing, calculators, drag and drop fields, and online procedure manuals. Workflow may include comparing orders in the system with system-generated collection reports, labels, or queues.

6.2 Data Entry

Validation of data entry is essential. It verifies that all input and output information retains its integrity, reliability, and accuracy. This validation is also strongly recommended to ensure that invalid or nonsense information will not be accepted by the system.

Prior to entering results, verify that all information pertaining to the specimen is complete and accurate. This would include:

• specimen identification number;
• specimen receipt date and time;
• bar-code or accession number;
• aliquot numbers or letters;
• tests to be run on the sample;
• if applicable, routing of the orders to correct instrument or laboratory section either by inquiry function or printed report; and
• verify reports that list samples that are pending.

Other data checks that are specific to the testing site should also be included in the validation process.

Test results should be entered during validation in the same manner as the normal laboratory workflow occurs, using all data entry functions available on the system, including worksheets, loadlists, and any back-up procedures (see the most current version of CLSI/NCCLS document LIS4—Standard Guide for Documentation of Clinical Laboratory Computer Systems). There should be documentation for testing of all critical steps. For example, tests performed on an instrument interfaced with the LIS should have the results entered into the system using the interface. Be sure to validate any tests that do not require a result, for example, procedures for workload documentation or that are used for generating charges. Quality control samples and results should also be included in the validation process.

Result values should test all defined parameters of the system and the specific test being validated. The defined parameters may include:

• system messages, warnings, and alarms;
• numeric boundaries: high, low, critical, and normal and reference ranges; linearity; delta checks; maximum and minimum allowable digits in result, decimal places; units of measure; age- or sex-specific values;
• alpha boundaries: all allowable entries; normal or abnormal;
• default values;
• results that include less-than, greater-than, or ratio symbols;
• changes to default values, if allowed by the system;
• procedures that may be ordered in intervals such as glucose tolerance panels;
• automated interpretations based on predefined result patterns, for example, ABORH (see the most current version of CLSI/NCCLS document LIS4—Standard Guide for Documentation of Clinical Laboratory Computer Systems);
• calculations based on one or more result entries;
• visual displays such as reverse video, highlighting, or changes in color of display fields;
• security between departments and user profiles;
• functionality that checks for duplicate test orders, system cancellations, or messages;
• speed typing functions for text entry such as templates or phrases;
• exception that may require a password or message acknowledgement; and
• rounding or truncation of results by the instrument interface.

Invalid data or nonsense information should also be entered into the data fields. This verifies that the system only accepts valid information. The invalid data entered should prompt any audible or visual fault alarms/warnings that are defined in the system. Some examples of invalid information are:

• entering alpha results in a field defined as numeric and vice versa;
• entering an alpha result other than the defined allowable responses;
• values that exceed the allowable number of digits for a test;
• diagnosis codes that are not valid for a particular test; and
• incorrect units of measure.

After data have been input into the system, inquiry functions should be used to verify that the data were accepted and stored correctly. It should also be verified that additional tests can be added to a specimen in various stages of processing. The following are some questions that can be asked to validate the accuracy of the input data:

• Can the results be accessed using the patient name or alpha identification?
• Can the specimen results be accessed using the specimen numeric or bar-code identification?
• Can all notes and text comments associated with the results be viewed?
• Do all tests, results, reference ranges, and units of measure for a specimen display as expected?
• Are numeric results in the correct format with the defined number of digits and decimal placement?
• If the numeric or alpha result is outside of defined parameters, is it obviously “flagged” as out of range?
• For corrected results, can the original and current result be accessed/viewed?12
• Is the status (collected, received, pending, completed) of the test(s) correct?
• Is the collection date and time stamp correct? Is the performed/verified date and time stamp correct?
• Do the ordered and resulted tests qualify for all designated loadlists and/or reports?
• Can the requesting doctor for a specific specimen be viewed?
• Did result-triggered expedited reports print to the designated printer?
• Can pending and in-progress/preliminary tests be identified and viewed?
• Can the samples and any corresponding aliquots be located and tracked for future analysis?
• For microbiology reports, do preliminary and final results display and are specifically defined as such?

• For microbiology reports, are sensitivity patterns displayed correctly and clearly?

Changes to the data input should also be performed and validated. These changes should include:

• corrections to all result types (numeric, alpha, text);
• amended and appended information;
• modifications to diagnosis codes that are attached to specific tests; and
• ordering additional procedures on a specimen.

6.3 Computer Calculations

Calculations, algorithms, and/or rules that are used in reporting patient results must be validated for accuracy and reproducibility. Every test procedure that is defined as a calculation must be validated. The calculated result should be compared to a manually calculated test result with the same data used in the computer calculation. Each calculated patient result should contain the desired number of significant figures for that specific test. The calculation formula, including rounding off or truncation rules used to calculate the results, should be included in the documentation. It is recommended that upper and lower result ranges be tested and that the result is reproducible. Calculations should be validated annually and when changes or revisions are made to the database and/or specific procedure.

6.4 Autoverification of Transmitted Results

Autoverification refers to any result transmitted over an instrument interface that is initially “evaluated” for pass or fail by the LIS, based on defined system parameters. If the value “passes,” then the result is automatically released by the system and is verified or completed. Any value that falls outside of the defined parameters “fails,” and must be reviewed by the designated operator. This guideline does not apply to electronic crossmatch. Validation should be performed at implementation and when any change is made to the system parameters.

Every test that qualifies for autoverification must be validated. It is recommended that samples or control reagents with known values be used for the validation process. If the instrument database will allow predetermined results to be entered for “test” samples and these results can then be transmitted over the interface to the LIS, this is acceptable for validation also. Developing a worksheet that defines each parameter for each test would be helpful in organizing the validation process. The following is a list of parameters that, when applicable to the test, should be verified. It is recommended that multiple values be used to validate each parameter:

• Results that fall within the parameters that allow the system to “pass” or verify the result to complete.

• Results that fall outside of the parameters that qualify the results as a “fail” and require human review and manual update to complete. Examples are critical high and critical low, high and low, high and low review limits, delta check fail, abnormal, invalid or zero value, and absurd values.

If the instrument has a database that performs an initial evaluation for pass or fail prior to transmission to the LIS, then this database should also be validated using these same guidelines.

Functionality or “rules” that prompt the system to automatically order confirmatory tests based on defined test results should also be validated.
The system must meet the defined criteria 100% for every test and defined parameter. It is recommended that the system be monitored daily, using a daily result log for example, to ensure the autoverification functionality is performing as expected.

6.5 Generated Reports

System-generated reports, as on-demand or by scheduled system operations, include any document used to identify samples and results. These documents may include instrument result reports, departmental worksheets used to organize workflow, worksheets used to review results prior to verification or acceptance, result review reports, workload reports, error reports, correction or exception reports, management reports, turnaround time reports, and patient/specimen chart reports.

6.5.1 Verification of Accurate Result Transmission

All information that is required on each report type by the testing laboratory/facility should be identified and validated for content and accuracy. Some examples are laboratory name, address, CAP and/or CLIA ID number, specimen identification number, location, date/time report generated, and spelling and name of ordering physician. The report format should also be validated, if applicable, for the correct headers, footers, column headings, page numbering, end-of-report messages, reference laboratory information, footnote/comment location, section dividers, page breaks or information that is “cut off” or incomplete, and the inclusive time interval of the report (daily, five day or seven day cumulative). QC reports such as Levy Jennings plots should be validated for accuracy. The printed report should be compared to a system inquiry function.

6.5.1.1 Correct Patient

Validate that the printed chart results correspond to the correct patient. Verify specimen ID number, location, and date of service.

6.5.1.2 Correct Physician

Validate that the ordering physician and if applicable, attending, referring, or consult physician are correct.

6.5.1.3 Results and Interpretive Data

Validate the result value, any alert flag (high, low, critical, review, delta check), decimal placement, significant digits, comments or footnotes, interpretive data, order date and time, date and time of service, units of measure, less-than and greater-than symbols, and charts and graphs.

6.5.1.4 Correct Reference Intervals

Validate that the correct reference intervals are printed, especially those specific to patient age, sex, or species (laboratories that perform testing on veterinary samples).

6.5.1.5 Corrections

Revised reported results must be accurate, clear, and specifically identified as such. Verify that all corrected results are identified and formatted per the facility’s established policy. Validate that both the correct and incorrect results are reported together, in sequential order, including any additional comments or notes. Validate that the correction date and time are documented.
6.5.1.6 Interfaces With Automated Instrumentation

Since a large amount of laboratory data contained in the patient medical record originate from an automated instrument interfaced with the LIS, it is recommended that this data transmission is validated. The validation process and test scripts/plan should be designed to ensure the interface is functioning properly and the design is accurate. Since not every automated instrument is the same, validation steps should be developed that are specific to each interface to be tested and specific to the site/laboratory. See the most current version of CLSI/NCCLS document GP19—Laboratory Instruments and Data Management Systems: Design of Software User Interfaces and End-User Software Systems Validation, Operation, and Monitoring for more information.

The following are some suggestions of validation points:

- Direction of data transfer; information uploaded to instrument, information downloaded from instrument;
- Specimen identification number entry; manual or bar-coded;
- Upload of patient demographics that may determine which sex- or age-specific high/low flags or reference ranges are downloaded with result;
- Upload of specimen type that may determine action messages or which reference ranges are downloaded with result;
- Mapping or cross referencing of instrument test codes versus LIS test codes; perform a one-to-one validation of each test upload and result download;
  - Ensure that the test ordered in the LIS uploads to the correct corresponding instrument test.
  - Ensure that the result downloaded from the instrument downloads to the correct corresponding LIS test.
  - Validate any instance in which there are multiple tests in the LIS that upload to the same test on the instrument.
- Result type flags; final, interpretation, preliminary;
- Result format; decimal placement, numeric rounding, alphanumeric, >,<;
- Warnings, error messages, and/or flags that determine if result is to be uploaded or not uploaded in the LIS; and
- Any site-specific custom settings.

6.5.1.7 Interfaces With Other Systems

If the LIS is different than the system used, by nonlaboratory personnel, to access patient information and results and/or the repository for the electronic medical record, these display data must also be validated.
It is recommended that the person(s) designated to validate the LIS be a trained user of the foreign system. If this is not possible, then there should be coordination from the foreign system staff to assist in this validation. The following data would be helpful in the validation process:

- a hard copy of the interface “message” transmitted to the foreign system from an LIS log file. (This will assist in identifying the exact data fields that should be populated in the foreign system database);
- print screens from the LIS;
- printed patient chart copies; and
- result review reports.

All applicable guidelines for data entry listed in Section 6.2 should be used in validation of the data displayed in the foreign system. Close attention should be given to results that include text comments, footnotes, and interpretive data associated with numeric results. If the registration data displayed in the repository originate from the LIS and not a separate EHRS interface, this information should also be validated for accuracy. Flags or symbols may not be universal between the two systems. In this case, validate that the conversion of the alert flag, between the two systems, is correct. For example, in the LIS, a high alert is an “H” beside the numeric result. On the repository display, the test name and numeric value font color is red. Other examples would be other font colors, bold, and reverse video.

All scenarios that would result in a transmitted result being “rejected” and qualifying for an error log should be tested. For example, if the two systems require the name, identification number, sex, and date of birth to be the same in order for a result to be updated to the repository database, then one or more of these required fields should be changed prior to the result transmission. Verify that the result did not update to the patient record and was routed to the error log. Coordinate with the foreign system staff to confirm the cause of the error, correct the error, and then resubmit the result message to the patient record. This process will validate that results with valid identification parameters only will be updated to patient records by the foreign system.

### 6.5.2 Retrieval of Data

It is critical that the LIS stores and retrieves data accurately. Data, whether on-line or archived, should be stored in a manner that prevents unauthorized access, modifications, or omissions. Regardless of the storage medium, there must be adequate space on the media for the archived data.

The system should be validated to ensure that it retrieves data on demand, accurately for the duration that the data are required to be stored/available on the system. It is recommended that printed reports or print screens, created prior to the data being archived or purged, be used to compare with the recovered data. The following are some questions that can be asked to validate data retrieval:

- Are archived or purged data retrievable, without difficulty, and within the facility’s established time limits?\(^{12}\)

- Are all numeric and alpha result values, decimal places, and format the same?

- Do the restored data include interpretive/diagnostic data; units of measure; and reference intervals that were reported with the original results?\(^{12}\)

- Can a hard copy of the original report be printed?
• If there have been revisions or upgrades to the system, can data archived or purged, prior to the updates, be restored and are they complete?

• Were there system messages or reports that would indicate that the records retrieved are corrupt or incomplete?

• If an archived record has been restored, corrected, and rearchived, is the corrected record plainly flagged as such?

All system messages that may indicate an error has occurred during data retrieval should be tested. Some examples are:

• Enter invalid or incorrect parameters during the retrieval process to prompt an error message or report.

• Attempt to restore a record that has not been archived or purged.

6.5.2.1 Updating Archived Reports

Validate that pending, amended, and corrected reports can be retrieved as an exact duplicate. Validate that any record that has been archived, restored, updated, and archived again, is clearly identified as such. Verify that any changes to the record do not obscure or omit any previous data.

6.5.2.2 Adequate Storage Capacity of Data

Verify adequate storage capacity of data by comparing the known storage capacity of the media with the size of the files that are to be copied to the media. If the file size is greater than the storage media capacity, then archive storage capacity is unacceptable. It is recommended that storage capacity be validated periodically after initial installation. This will ensure accurate and complete archived data as the volume of data to be archived increases over time.

7 Quality Assurance Standards/CQI

Laboratory Information Systems have become essential tools in the diagnostic testing process. Their validity must be assessed before implementation just as laboratory instruments are evaluated. The validation process ensures that all components of the LIS, including the hardware, software, and peripheral devices, function properly for their intended use.

The institution should select the LIS based on established supplier qualification criteria. The criteria should ensure that the potential LIS meets the defined requirements. The requirements may include intended-use, regulatory, hardware, integration, implementation, maintenance, user-interface, financial considerations, and any other requirements. The user at the individual institution must fully validate the system before implementation. In some cases, the vendor may provide guidance for end-user validation, but the ultimate responsibility to meet the validation’s goal is still the responsibility of the user. The system must be validated using the institution’s own standard operating procedures, database elements, specific requirements, etc., and the results must be documented.

7.1 User Responsibilities

• Validation plan/protocol.
• Determine types of testing.
• Develop test cases.
• Document testing results.
• Document hardware and software change controls.
• Document evaluation and acceptance.
• Document personnel participating in validation activities.
• Document SOP reviews, training plans, installation, and implementation.
• Document review and approval authority and dates.
• Perform periodic system evaluation and maintenance.
• Revalidation as necessary.

A comprehensive LIS validation plan with clearly stated goals and strategies is needed to realize success. Software verification and validation employs review, analysis, and testing techniques to determine whether a software system and its intermediate products comply with requirements. These requirements include both functionality capabilities and quality attributes such as accuracy, completeness, consistency, safety, and security. An LIS validation plan can be approached systematically, starting with system identification, system definition of physical properties, interfaces and peripheral devices, and general functionality.

Table 1. Validation Plan. The LIS validation plan should describe documents, hardware and software verification and validation (V&V) tools, techniques, methods, and operating and test environment to be used in the V&V process.

<table>
<thead>
<tr>
<th>Effective Date:</th>
<th>LIS ID:</th>
<th>Version:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Prepared by:</td>
<td></td>
<td>Date:</td>
</tr>
<tr>
<td>Reviewed by:</td>
<td></td>
<td>Date:</td>
</tr>
<tr>
<td>Approved by:</td>
<td></td>
<td>Date:</td>
</tr>
</tbody>
</table>

**Scope:** Define the scope of the validation plan that is intended to be followed in testing a specific portion of the LIS or the total system in order to produce documented evidence that the system operates consistently and reliably meets its predetermined specifications and intended use.

**Purpose:** What the purpose of the validation is. Full prospective validation of a new system or limited revalidation due to changes.

**System specifications:** The specifications should describe the functions the system is required to perform and expected performance criteria, system limitations and constraints, and any system faults.

**Testing environment:** Describe the environment in which testing will occur.

**Risk analysis results:** Identify, evaluate, and mitigate potential problems.

**Method:** Describe the methods and procedures for each task, including online access, and conditions for observation/evaluation of development processes. The plan shall define the criteria for evaluating the task results.

| Responsibility: Identify the organizational elements or individuals responsible for performing the V&V tasks. | Resources: Identify the resources for the performance of the V&V tasks. | Contact Information: Name, location, and telephone number | Timeline: Describe the schedule for V&V and the target dates for completion. |
### Table 2. LIS Identification Comment

The purpose of this form is to have a current list of LIS inventory computer hardware, software, and ancillary for easy configuration management. The laboratory does not have to “search” for what it currently has. Changes are in Controls.

<table>
<thead>
<tr>
<th>Computer hardware</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Manufacturer, model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Processor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Internal memory (RAM)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Graphic adapter</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hard disk (partitions, memory sizes)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Installed drives</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Pointing device (e.g., mouse)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date of installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Monitor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufacturer, model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date of installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Printer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufacturer, model</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Serial number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Space requirement</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date of installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Instrument interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type, select code, slot number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date of installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Contact information</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Connected equipment hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware module 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Interface card setting</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Date of installation</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Network components</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Type/function</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Hardware</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating software</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Operating system (version)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>User interface</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application software 1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufacturer/vendor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product number (version)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required disk space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release date</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Application software 2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Description</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Manufacturer/vendor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Product number (version)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Required disk space</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Release date</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
The objectives of the validation efforts are to find defects and to determine if required functions and attributes are built into the LIS. The primary goal of the software validation is to demonstrate that the complete software end product complies with established software and LIS requirements. Software validation activities should be conducted throughout the entire software lifecycle and they do not end until the software is no longer used.

7.1.1 Why Validate?

- Regulatory and standards requirements.
- Patient safety.
- Ensure that all elements of a system (hardware, software, peripheral devices, personnel, documentation, and environment) result in expected performance (accurately and reproducibly).
- Good business practice: find problems and resolve problems.
- Legal: liability coverage.

Verification and validation (V&V) should be performed for the life of the software; the initial effort may be completed with the installation and checkout of the software system. A final report should be prepared to summarize the activities and results of the V&V. This report serves several purposes:

- It serves as the starting point for V&V of the maintenance of the system.
- It can show lessons learned to improve the V&V process for the next project.
- It documents outstanding unresolved anomalies from development, installation, or operation.

During operation and maintenance, any modifications, enhancements, or additions to the software system must be verified and validated. The level of V&V effort is dependent on hazard/risk analysis results.

How much validation is needed? LIS validation allows the user to have a high level of confidence that the system or specific process will consistently produce a product meeting its intended use and predetermined specifications. The level of confidence needed is therefore proportionally equivalent to the level of LIS validation required. Verification and testing effort will vary depending upon the safety risk posed by the automated functions of the system. Risk analysis will help to determine the scope and the degree of LIS validation. The validation effort should be commensurate with risk. A system associated with critical devices or components that could result in direct and severe patient harm should require a more extensive validation effort. A system that could result in minor consequences could have simple verification and monitoring to confirm operation against expected values.

There are various risk analysis tools available. The facility should evaluate and use the tool that is best suited for its application. The common questions to ask when performing risk analysis are:

- What happens if this process fails?
- How can it fail?
- What risk is involved if it fails?
- What can we do to prevent it from failing?
- Will injury or death to the patient result if the system fails? (clinical risk)
• Has financial or business risk been considered? (property or monetary levels or customer services)

• What is the effect of system or process failure on patients, operators, bystanders, service personnel, and the environment?

The more critical the requirement is to product or process quality or for safety, the more detailed and rigorous the specifications and associated testing should be.13-15

Table 3. Definition of Severity of Clinical Risk16

<table>
<thead>
<tr>
<th>Minor Level</th>
<th>Failures or latent design flaws would not be expected to result in any injury to a patient, operator, or bystander.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Moderate Level</td>
<td>Failure or latent design flaws will directly affect a patient, operator, or bystander so that it could result in nonserious injury. Operating the software will indirectly affect the patient, operator, or bystander so that incorrect or delayed information could result in nonserious injury.</td>
</tr>
<tr>
<td>Major Level</td>
<td>Failures of latent design flaws will directly affect a patient, operator, or bystander so that it could result in death or serious injury. Operating the software will indirectly affect the patient, operator, or bystander so that incorrect or delayed information could result in death or serious injury.</td>
</tr>
</tbody>
</table>

7.1.2 Testing

Testing is the major activity of validation efforts. It is intended to challenge the application software and other parts of the overall system, functionally and structurally. The number and types of tests performed will depend on the criticality of the function of the software application or hardware function.13-15

Consider the following when using testing tools:

• The testing environment should be as true to the production environment as possible.

• The test case should address all critical functions of the software and hardware and have a reasonable probability to detect a failure or flaw.

• Design test cases to verify and challenge the user’s requirements, the system specifications, the user’s operation manual, and the technical or maintenance manuals.

Consider the following when developing test designs:

• required features to be tested;
• specified quality attributes;
• load limits;
• stress tests;
• configurations that the software may need to support;
• compatibility with existing or planned system components;
• security of the system;
• storage limits of the software, and related data;
• performance (accuracy) response times, and throughput;
• installability, primarily in the user environment;
• reliability/availability to the specifications;
• recovery from software and data failures;
• serviceability requirements;
• user’s guides;
- human factors for usability and acceptability;
- interfaces with other system components; and
- hardware interfaces.

Table 4. Testing Type

<table>
<thead>
<tr>
<th>Test Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Normal Testing</td>
<td>Testing the functional and structural integrity of the system. Demonstrating that the system performs as intended under normal conditions. Input within-range data.</td>
</tr>
<tr>
<td>Boundary Testing</td>
<td>Using test values that are set at the edges of valid input ranges to force the system to make a decision as to which branch of the program to execute or whether the input is valid or invalid.</td>
</tr>
<tr>
<td>Invalid Case Testing</td>
<td>Use invalid inputs to demonstrate system checks for input errors.</td>
</tr>
<tr>
<td>Stress Testing</td>
<td>Using input values at the limits of system-specified requirements or worst-case scenario to challenge the computerized system, peripheral devices, and/or components at their physical limits and document their continued ability to perform correctly.</td>
</tr>
<tr>
<td>Special Case Testing</td>
<td>Using input values that are most likely to cause program errors to test how the system responds to unsuitable data. Using input values of specific types of data or no data entry in a field will help to test the system defaults.</td>
</tr>
<tr>
<td>Parallel Testing</td>
<td>Testing the systems as a whole by running two systems in parallel and comparing the outputs. Parallel testing is the most common type of testing, but it cannot be relied upon as sole criteria for validating the system because it focuses on a narrow range of values during normal test runs.</td>
</tr>
<tr>
<td>Vendor Demo/Alpha or Beta Testing</td>
<td>This type of testing is not acceptable as a substitute for validation by the end-user. The system’s operation and testing at the time of installation should be performed by the end-user at each site.</td>
</tr>
</tbody>
</table>
**Table 5. Test Cases for Each Function.** Test cases should include boundaries, high load/stress, invalid, and special case testing.

<table>
<thead>
<tr>
<th>Test case number:</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Test case type:</td>
<td>□ Normal □ Boundary □ Invalid □ Stress □ Special case</td>
</tr>
<tr>
<td>Specification:</td>
<td></td>
</tr>
<tr>
<td>Purpose of test:</td>
<td></td>
</tr>
<tr>
<td>Test environment:</td>
<td></td>
</tr>
<tr>
<td>Test procedure:</td>
<td></td>
</tr>
<tr>
<td>Step 1:</td>
<td></td>
</tr>
<tr>
<td>Step 2:</td>
<td></td>
</tr>
<tr>
<td>Step 3:</td>
<td></td>
</tr>
<tr>
<td>Step 4:</td>
<td></td>
</tr>
<tr>
<td>Expected result:</td>
<td></td>
</tr>
<tr>
<td>Acceptance criteria:</td>
<td></td>
</tr>
<tr>
<td>Actual result:</td>
<td></td>
</tr>
<tr>
<td>Comment:</td>
<td></td>
</tr>
<tr>
<td>Severity Risk Analysis:</td>
<td>□ Minor □ Moderate □ Major</td>
</tr>
<tr>
<td>Test performed by:</td>
<td>__________________________</td>
</tr>
<tr>
<td>Reviewed by:</td>
<td>__________________________</td>
</tr>
</tbody>
</table>

**7.1.3 Process Validation**

In addition to software testing, the end-user should ensure that the hardware including all peripherals and system software is calibrated and installed in accordance with the manufacturer’s instructions.

**Table 6. Elements of Process Validation**

<table>
<thead>
<tr>
<th>Installation Qualification (IQ)</th>
<th>Establishing by objective evidence that all key aspects of the process equipment and ancillary system installation adhere to the manufacturer’s approved specification and that the recommendations of the supplier of the equipment are suitably considered. (GHTF Guide, 2.1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operational Qualification (OQ)</td>
<td>Establishing by objective evidence process that equipment and ancillary systems are capable of consistently operating within predetermined requirements.</td>
</tr>
<tr>
<td>Performance Qualification (PQ)</td>
<td>Establishing by objective evidence that the process and product under anticipated conditions consistently perform safely, effectively, and reproducibly, and meet all predetermined requirements.</td>
</tr>
</tbody>
</table>
7.1.4 Revalidation and Change Control

Changes or modifications and/or enhancements made to the hardware, software, peripheral devices, and any other critical component of the LIS should be evaluated for risk, the impact of the change, and an assessment made of the level of validation required. Records of changes should be documented and maintained in accordance with established protocol. At a minimum, the change control records should include:

- a description of the change;
- a description of the tasks performed to effect the change;
- the date of the change;
- the person making the change;
- equipment, modules, process, or functions that are affected by the changes;
- authorized signature and date;
- validation protocols;
- validation results; and
- documentation of approval and acceptance.

7.1.5 Event That May Require Revalidation

The user should perform a risk analysis when changes occur to determine if revalidation is needed. Risk analysis is done to identify the critical process, critical operating, and performance parameters that are affected by the changes and require validation efforts. Some changes that may require validation include:

- new software version;
- patches;
- change in standard operating procedure;
- change in functionality;
- new hardware;
- interface change;
- integration of database; and
- recovery from a computer crash or natural disaster, such as fire, flood, or earthquake, that corrupted critical data.

NOTE: This is not intended to represent a comprehensive list of events that require revalidation.

7.2 Database Maintenance

The maintenance process is activated when the software product undergoes modifications to code or associated changes caused by a problem or a need for improvement or adaptation. Modifications of the software must be verified and validated, and software integrity level assignments should be assessed during the maintenance process. The software integrity level assignments should be revised as appropriate to reflect the requirements of the maintenance process. These modifications may be derived from requirements of the maintenance process, from requirements specified to correct software errors (e.g., correction), to adapt to a changed operating environment, or to respond to additional user requests or enhancements.

7.2.1 Safeguarding Data and Data Maintenance by Verifying Data Integrity

The database maintenance process should include a Database Maintenance Plan for the entire system. Additionally, a database back-up and a solid Disaster Recovery Plan for the database should be created.
Questions that need to be addressed are:

- In case of a catastrophic event, is it certain that the laboratory can recover?
- How long will it take to recover and have the system available for normal operation?
- How much data loss can the laboratory tolerate?
- How and where can one get the required hardware, the configuration of the servers, and service pack information?
- Who is to communicate what, who are the people to be contacted in the event of a disaster, and how are they to be contacted?
- Are there procedures clearly defined in your SOP for correcting discrepant data or duplicate data?

7.3 **Certification of Interfaces**

Verify the correctness, accuracy, and completeness of the system requirements, allocation to hardware, software, and user interfaces against user needs.\textsuperscript{13-15}

1. **Correctness**: Verify that performance requirements such as timing, response time, and throughput allocated to hardware, software, and user interfaces are sufficient/adequate to satisfy user needs.

2. **Accuracy**: Verify that the internal and external interfaces are specific to the data formats, interface protocols, frequency of data exchange at each interface, and other key performance requirements to demonstrate compliance with user requirements.

3. **Completeness**: Verify that application-specific requirements such as functional diversity, fault detection, fault isolation, and diagnostic and error recovery satisfy user needs. Verify that the user’s maintenance requirements for the system are completely specified. Verify that the migration from the existing system and replacement of the system satisfy user needs.

   a. Functionality (e.g., algorithms, state/mode definitions, input/output validation, exception handling, reporting, logging);

   b. Process definition and scheduling;

   c. Hardware, software, and user interface descriptions;

   d. Performance criteria: timing sizing, speed, capacity, accuracy, precision, safety, and security;

   e. Critical configuration data;

   f. System device and software control, initialization, transaction and state monitoring, and self-testing.

**Interface Control**:

1. What are the organizational interfaces?
2. What are the important interfaces between adjacent phases of the life cycle?
3. What are the interfaces between different entities of the computer programs?
4. What are the dependent hardware interfaces?
5. Where are the documents defined and maintained that are used in interface control?
6. What are the procedures for making changes to these interfaces’ documents?

Three interface areas:

1. User interface: required screen format, protection mechanisms, page layout, and content of the reports, relative timing of inputs and outputs, etc.

2. Hardware interface: identify electronic devices, firmware, communication devices, and output devices. Also identify applicable standards for these interfaces and verify the current application’s interface.

3. Software interface: identify the data management system, operating system, or library package, and interface with the other application. Verify correct software interfaces to them. Consider the following:
   - Are the interface objectives technically adequate and well-understood?
   - Are all data elements well-defined?
   - Are all restrictions and constraints clearly defined?
   - Were all the different kinds of interfaces taken into account and properly described?
   - Are all hardware-to-software functional interfaces specified in quantitative terms such as units, bits per second, message formats, priority rules, word length, timelines, and protocols?
   - Are all software-to-software interfaces functional, and are all data interface levels specified in quantitative terms, such as data timeliness, data definition, data formats, priority rules, message content, and communication protocols?
   - Are the interface performance requirements well-defined, and are the limits specified?
   - Is the criticality of the interface taken into consideration?
   - What are the impacts if the interface is degraded?
   - Are the interfaces testable and maintainable?
   - Have all appropriate standards been identified for the interfaces, including those from the current software application to its environment?

7.4 Problem Reporting and Tracking

The package software and any required database must be installed in the target environment according to the package instructions. The installation report should document the installation and any problems that are encountered.13-15

During operation and support activities, and support requests, maintenance activities are logged, tracked, and maintained. Data from feedback are collected and analyzed. Anomalies are identified and reported. A procedure must be established to describe the practices to be followed for reporting, tracking, and resolving problems identified in both software items and the software maintenance process. The
procedure should state the specific organizational responsibilities concerned with their implementation to verify and validate the use of problem reporting and corrective action practices and procedures. Problems encountered during software operation or maintenance may result from defects in the software, supporting and development process, hardware, or operations. Because of their diversity, the determination of the sources of a problem and the appropriate corrective action require a systematic control for monitoring problems and determining root causes. The purposes of problem reporting and corrective action systems are to:

1. Ensure that problems are documented, corrected, and used for process improvement.
2. Ensure that problem reports are assessed for their validity.
3. Ensure reported problems and their associated corrective actions are implemented in accordance with approved solutions.
4. Provide feedback to the vendor and the user of problem status.
5. Provide data for measuring and predicting software quality and reliability.

Problems can be resolved through corrections or enhancements. Corrections are documented. A Report Log should be maintained to ensure that all problems are tracked until they are resolved and the resolution has been approved.

The Report Log should contain the following information:

- what the anomalies are;
- source and cause of product or process problem;
- product(s) or process(es) presumed to contain the error, including documentation;
- problem severity;
- course of corrective action;
- impact on customer, cost, schedule, and risk; and
- problem/modification identification, classification, and prioritization of software must be identified from the following maintenance types:
  - Corrective: maintenance performed to correct faults in hardware or software.
  - Adaptive: software maintenance performed to make a computer program usable in a changed environment.
  - Perfective: software maintenance performed to improve the performance, maintainability, or other attributes of a computer program.

A procedure should be established to describe the method of reporting and resolving anomalies, including the criteria for reporting an anomaly, the anomaly distribution list, and authority for resolving anomalies.
Table 7. Problem Report Form

<table>
<thead>
<tr>
<th>Laboratory Name</th>
<th>Confidential</th>
<th>Problem Report #</th>
</tr>
</thead>
<tbody>
<tr>
<td>LIS ID</td>
<td>Release date</td>
<td>Version</td>
</tr>
<tr>
<td>Report type:</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Suggestion</td>
<td>Software</td>
<td>Hardware</td>
</tr>
<tr>
<td>Severity of problem:</td>
<td>Minor</td>
<td>Moderate</td>
</tr>
</tbody>
</table>
(If the severity of the problem is major, additional reporting to a regulatory agency, such as FDA for Medical Device Reporting, may be required.)

Module program used:

Accession number:

Description of problem: Attachments (Y/N)

Can the problem be repeated? (Y/N)

Action taken: By: Date:

If reported to the vendor, reported date, and vendor problem reference number.

Resolution:

Reported by: Date:

LIS manager/QA review and approval Date:

7.5 Train Staff

Personnel need to be trained to install, validate, operate, and maintain the software. It is essential that the training be done before implementation of the software package. Personnel who perform verification and validation activities must be made aware of defects and errors that may be encountered as part of their job functions. Personnel must also be assessed for competency prior to implementation of the LIS.

Table 8. LIS Training Checklist

<table>
<thead>
<tr>
<th>Training manual</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Competency assessment</td>
<td></td>
</tr>
<tr>
<td>Password and electronic identification</td>
<td></td>
</tr>
<tr>
<td>Computer security</td>
<td></td>
</tr>
<tr>
<td>Confidentiality</td>
<td></td>
</tr>
<tr>
<td>LIS overview tour</td>
<td></td>
</tr>
<tr>
<td>User’s Manual and related LIS documentation</td>
<td></td>
</tr>
<tr>
<td>Change Control Procedures</td>
<td></td>
</tr>
<tr>
<td>Specific operational and functional training</td>
<td></td>
</tr>
</tbody>
</table>
Table 9. System Implementation Checklist

| Implementation Plan                        |                          |
| Schedules and coordination of resources   |                          |
| Assign responsibilities and personnel     |                          |
| Develop training plan/manual materials   |                          |
| Schedule training                         |                          |
| Develop SOPs                              |                          |
| QA review of records (validation results, training requirements, problem reports) |                          |
| Develop contingency plan                  |                          |
| Develop postimplementation monitoring     |                          |

7.6 Postimplementation Monitoring

As part of the Quality Assurance program, the user should develop a clearly defined process to routinely monitor computer systems. The QA monitor provides confidence that all systems and components that influence the quality of the product are working as expected. Points to consider include:

- All hardware and software have been subjected to appropriated calibration, validation, and routine maintenance.
- Documentation regarding the LIS and components is present and complete.
- Vendor upgrades, repairs, and replacement of the components have been evaluated for their impact on operations.
- Appropriate training and competency assessment have been performed.
- Software, hardware, and peripherals devices/components are being used in accordance with applicable manufacturer’s instructions, SOPs, and specific requirements.
- Software, hardware, and all peripherals devices/components are operating as specified and no unauthorized changes have been made.
- SOPs, change controls, system maintenance, and data integrity are followed.
- System malfunction and problems have been appropriately investigated, evaluated, and addressed.

7.7 Documentation

All LIS-related documents and records should be reviewed for accuracy and completeness and should be retained for the life of the system or in accordance with established record retention protocol. Records should be available to accreditation or regulatory agencies upon request.
References


12. CAP. Laboratory General Checklist. College of American Pathologists, Commission on Laboratory Accreditation Program; 2003.


Additional References


Appendix A. Software Design Recommendations for Manufacturers

Laboratory information management systems (LIMS) are Class I devices (21 CFR 862.2100, Calculator/Data Processing Module for Clinical Use). They are included in the category of electronic medical devices intended to store, retrieve, and process laboratory data. LIMS may also handle scheduling, billing, and other nondevice functions. LIMS have been exempted from 510(k) since 8 June 1988. However, compliance with all other requirements is required, including registration, listing, Good Manufacturing Practices (GMP) or Quality System Regulations (21 CFR 820), Medical Device Regulations (MDP) (21 CFR 803), and Correction and Removal Reporting (CAR) (21 CFR 804).

NOTE: If the LIMS only receives information for the patient record and neither transmits any instructions to any interfaced medical device, nor controls any medical device functions or alarms, a premarket notification is not required.

The LIMS exemption does not apply to applications of artificial intelligence or other algorithms intended to assign a probability of diagnosis for the purpose of guiding therapy or further diagnostic studies. Such clinical data management functions may be subject to additional FDA regulations as are blood establishment software systems.

The Design Controls requirement is only one part of the overall requirements for LIS. End-users should perform vendor qualification and verify that LIS vendors comply with regulatory requirements.

A1 Software Development and Design Validation Methods

Validation is expected to permeate the product lifecycle regardless of development methodology or lifecycle model. Software must provably meet requirements and demonstrably be implemented correctly and completely. This chapter provides an overview of design controls, software design, and validation tools for use by system designers and implementers. This section supplements NIST special publication 500-234 by elaborating on specific software validation methods for common laboratory information system components like communication protocols and expert systems.

A1.1 Design Controls

Quality and technical standards such as ISO 9001 provide a way to demonstrate that a system is developed and transferred in a controlled manner. FDA CFR 1997 references the ISO 9001 and ISO/DIS 13485 and provides a good discussion of design control throughout a product’s lifecycle, encompassing planning, design input, output, review, verification, validation, transfer, and change. Planning includes specifying and reviewing the development activities and responsibilities for design input, development, and implementation. Design inputs are expected to draw from technical and business expertise and include functional, performance, and interface requirements. Requirements should be quantifiable and testable. Technical expertise can help decide the feasibility of requirements and business expertise can help qualify the expected benefit gained from requirements. Design outputs are materials that specify and describe the design, including technical specifications and documentation, and typically provide input to refining iterations of design and development. Additional design outputs include quality assurance procedures and installation instructions. Design review is a documented examination of design to evaluate the requirements, the design, and potential problems. Verification documents that the design output matches the design input requirements. A “traceability matrix,” for example, can help show how design output like software modules meet design input requirements in a tabular format. Validation ensures that the system conforms to user needs and intended uses by testing, and includes software validation and risk analysis.

Design transfer for a software product involves communicating necessary specifications from the development team to the configuration management team to build the product correctly. A configuration
management system can also help ensure design change occurs in a controlled manner. As with other activities, design change should be documented. Appropriate procedures and documentation should be provided when accepting a system, when evaluating an existing system, or when changing a system to make sure the system is suitable for its intended purpose.3

A2 Software

A2.1 Design

Best Practices

Regardless of development methodology, modeling technique, or implementation technology, some guidelines for development follow.

Software design should adhere to widely accepted principles and standards. Often the designer must meet multiple objectives in terms of different criteria, and choose to uphold one set of techniques or principles over another, so a good practice is to document the reasons for design decisions. Aside from providing accountability, should conditions predicated a decision change, future development iterations may overturn the decision and provide specific counterarguments.

Architecturally, information systems are well-served by implementing tiered functionality and loosely coupled layers of service.

Using existing specifications from international standards can help reduce development and theoretical validation efforts, since existing materials are often readily available.

Validation

The remainder of this section is devoted to validation methods as applied to software design, and is intended for use by laboratory information system implementers (i.e., vendors that design the software, users who configure expert systems). Laboratory information systems are often comprised of components built, configured, and deployed by different parties at different times. Since in practice, some system software components may be developed in-house, design phase validation techniques are provided to assist developers. Interfaces by nature may involve multiple parties, so protocol validation techniques are provided for use by architects in the design phase, and for reference by those who accept the composite system during the acceptance phase. A system component may provide a framework which allows the end-user to influence system behavior through data configuration, module plug-ins, or a high-level language for expressing facts or rules, so logical validation techniques are provided for the power-user who expresses the desired behavior and the party responsible for accepting the changes.

Formalism provides a mathematical language for reasoning about a model. Given a formal specification for a model, algorithms can be used to verify or improve the quality of a static model, and manual proofs can be constructed and automated validation techniques can be applied for a dynamic model. The validation methods surveyed in this section are intended as informative examples for common types of models.

Like other software, laboratory information systems can use static models to describe and review modules and data design, but in particular may need to make use of formal validation techniques with relation to communication protocols and knowledge bases. Proof of correctness can be provided using dynamic
Appendix A. (Continued)

mathematical models to describe and validate system behavior, and first-order logic can describe and validate an expert system.

A2.1.1 Modules

During development, static modeling of modules and data may be checked for quality in terms of software engineering principles by review and inspection.

A variety of metrics and rules of thumb for module and function design are available, and designers can reason to uphold specific principles.

Modularity, for example, reduces development time and simplifies validation. Modular design can be qualified in terms of “coupling,” the interdependency of modules, and “cohesion,” the strength of the relationship between elements in a module.

The concepts of coupling and cohesion can be extended to qualify the design of object-oriented systems. Eder, Kappel, and Schreft provide a comprehensive, qualitative taxonomy where coupling for object classes can be thought of along interaction, component, and inheritance dimensions and cohesion along method, class, and inheritance dimensions. Interaction coupling represents relationships by invocation or data sharing, where communication by relevant parameters is preferred to global or external sharing. Component coupling represents use of one class by another in its structure or methods. In inheritance coupling, extension of a concept is preferred to redefinition of behavior by subclasses.

Method cohesion represents the relatedness of elements of methods, class cohesion represents the degree to which elements within a class are related, and inheritance cohesion represents the degree to which the inheritance hierarchy represents conceptual generalization rather than code sharing.

A popular qualitative, stylistic rule for object-oriented design is the Law of Demeter proposed in Lieberherr and Holland, which suggests that a method for an object should be restricted to operate on its arguments, the data and methods immediately contained by the object and objects created by the method, and that the directly interfacing classes are preferred for access from a class’s method.

A2.1.2 Data

While a number of data modeling techniques are available, popular techniques are relational data modeling, object-oriented design, and the use of declarative predicate logic.

Relational Model

The relational model is a popular data model based on set theory and was introduced by Codd in 1970. A relation schema is a set of attributes that each map to a set of atomic values in a domain, and a relation is a set of “tuples,” sets of \( \langle \text{attribute}, \text{value} \rangle \) pairs that map attributes to domain values.

Inconsistency in a relational model can be reduced by using a normalization process, which provides a way of decomposing relations based on functional dependencies of attributes. A functional dependency between attributes is a semantic relationship determined as part of modeling, and rules of inference may be applied to refine these dependencies. While “normal forms” can be used to qualify design, further desirable qualities based on functional dependencies of attributes are that dependencies are preserved in any decomposition and that information should not be lost by producing spurious tuples in a “natural join” of relations by attributes. Discussion and algorithms for decomposition and preservation of these
Appendix A. (Continued)

qualities can be found in Elmarsi and Navathe, 6 as well as proofs for the validity of inference rules for refining dependencies.

Object-Oriented Design

Object-oriented modeling represents a data structure and its associated operations as a “class,” and allows for reuse and specialization by allowing classes to inherit the structure and operations from ancestor classes.

Design patterns can help reduce development time by providing solutions known to have met with success. While the concept of design patterns as a way to communicate a solution to a known type of problems is not specific to object-oriented design, object-oriented design can greatly benefit from patterns due to the dimension introduced by inheritance.

The Demeter kernel model presented in Lieberherr and Xiao 7 is an example of a formal foundation for representing the structural aspects of the data model using class dictionary graphs and object graphs in terms of class dictionary graphs. Class dictionary graphs represent class definition and are defined in terms of “alternation edges” representing inheritance and “construction edges” representing containment. Object graphs describe object construction given a class dictionary graph. For the Demeter kernel model, a class dictionary graph must be free of cycles in inheritance and all inherited structural components should be uniquely named. In the inductive version of the model, each class used in the object graph should have a base case for its definition. The inductive class dictionary graph can be topologically sorted and new objects can be incrementally added as long as every potential circular definition has a noncircular base case for object construction. In general, reducing requirements for circular definition should help ameliorate object construction and consistency implementation issues.

The normalization concepts used in relational modeling can be extended to object-oriented design to improve cohesion and reduce data redundancy, and so reduce the risk of update or deletion anomalies. A formalization and method for reducing data redundancy in object-oriented databases is introduced in Hong, 8 where “abstractions” are thought of as entities to be refined into classes to produce a more “well-defined form.” Common or cross-linked attributes and methods are removed by the second form, and a single abstraction remains for the class in the third form. Attribute dependencies, where attributes always appear together, can be thought of as multivalued dependencies and so the axioms and rules of inference for the relation model can be applied to classes. A method is a function of a class uniquely defined by its name, input arguments and read attributes and produced output arguments, changed attributes, and initiated messages. A method dependency is said to exist for attributes that are read or written to by a method. Rules of inference apply for this type of dependency as well. Attribute and method dependencies together define abstractions for which the rules of inference may be applied to normalize toward a more well-defined form. The approach for applying normalization is to normalize classes first in order of hierarchy from the highest superclass down, then for object reference chains for inner references, and finally the remaining classes and the global result. The global normalization process deals with redundancy between classes by resolving naming conflicts so that the same name represents the same data and functionality and by improving cohesion so that redundancy recognizable by class and method similarity is reduced.

Formalisms like state machines can be used to model and validate object behavior. Further discussion of behavioral modeling and validation is given in Section A2.1.3.
Appendix A. (Continued)

Predicate Logic

Predicate logic is a technique for knowledge representation where objects are described with relations called predicates. Facts can be specified like relations in the relational model, and the meaning of an attribute value can be determined by its position. Rules are like relational views formed by applying inference procedures on the facts.6

Predicate logic can be normalized to a form consisting of intersections of atomic facts implying unions of conclusions, for which an inference procedure called “resolution” is sound and complete.9 Further discussion of validation for systems expressed in predicate logic is given in Section A2.1.4.

For the sake of efficiency, normalization procedures also exist for rules in deductive object-oriented databases. Xie and Han10 introduce a technique for flattening nested rules, composing attributes for an object into the same object “molecule,” and making an explicit class membership between an object and class so that the resulting object molecules can be thought of as predicates.

A2.1.3 Behavior

A system can be dynamically modeled with an appropriate mathematical formalism to validate its behavior. State machines are a mathematical formalism that represents the idea that a system moves between distinct static representations when events occur. Finite state machines, Petri nets, and evolving algebras can be used to provide a way to mathematically specify, reason about, and validate software behavior. Behavior and data can be modeled together with high-level extensions of finite state machines like “communicating finite state machines,” and of Petri nets like “colored Petri nets.”

Communication protocols are distributed algorithms that need to be provably free of concurrency problems like race conditions and deadlocks. By design, algorithms may avoid deadlocks or detect and preempt deadlocks, so that the hold-wait, circular wait, mutual exclusion, and no-preemption conditions do not hold. Livelocks, however, are a scenario where lack of progress is possible due to incompleteness. Both types of cycles can be detected by performing a state-space search on a validation model. The search space may be reduced by providing induction proofs about behavior in terms of system invariants, by using reduction where layering preserves behavioral independence, or incremental composition methods where highly dependent state machines are combined to preserve the external behavior of the systems.11

Finite State Machines

Finite state machines can be thought of as directed graphs where nodes represent states and arcs represent transitions. The representation facilitates demonstrations of completeness. Reachability analysis can be thought of as finding a transitive closure, and algorithms can be used to minimize and combine finite machines to help simplify design and validation.

In a network of “communicating finite state machines,” machines communicate using message queues that map the output of one machine to the input of another.

An extension of these machines is to additionally include variables, transfer only integer values in the queues, and allow operations to manipulate the variables. Variables can be used in conditions that guard transitions and transitions can assign variable values.

Gerard Holzmann introduced a language called PROMELA for validation models and uses this extension for communicating finite state machines.11 PROMELA represents “temporal claims” by providing special statements for “assert,” “accept,” and “never.” An “assert” clause can be used to express system
Appendix A. (Continued)

invariants in a monitor process. A “never” clause with respect to a finite state machine describes invalid state sequences to be checked in lock step against every transition in the validation model. In this way, liveness and safety terminating conditions can be checked. Livelocks can be detected by using the “accept” clause to specify states that may not be repeated infinitely inside a “never” claim.

Petri Nets

A Petri net is a directed, bipartite graph with two types of nodes: states, or “places,” and the “transitions” between the states. Arcs represent the relationship of places to transitions, and transitions to places. Mathematically, an incidence matrix can be used to represent these relationships.

“Tokens” are used to mark places and represent the state of the entire system, and so multiple state systems, for example, processes and communication channels, can be represented simultaneously without introducing composite states. Mathematically, a state vector can be used to represent the placement of tokens. Events can indicate the placement of tokens and so enable transitions for the marked places. When a transition fires, the token is removed from the original place, the precondition, and tokens are placed at successor places, the postconditions for the transition. Abstraction is supported in that places or transitions can represent a subsystem also modeled as a Petri net.

A colored Petri net is a type of high-level Petri net that allows tokens to be distinguishable data values, represented with colors in a colored Petri net, and these tokens may belong to data types or color sets. For a transition to occur, variables are bound to values in their types so that the arc expression, or inscription, for each incoming arc evaluates to a token value present at the corresponding precondition. Colored Petri nets can be extended to represent object-oriented design where tokens represent instances of classes and arcs select tokens according to their class.12,13

Validation proofs can be constructed using Petri nets and proof techniques like assertional reasoning and induction. When validating a distributed algorithm like a communications protocol, temporal propositions, that a condition always holds or eventually will hold, may be useful constructs for proving the absence of deadlocks and “liveness” of a protocol. By proving certain behavioral properties, an exhaustive state-space search may be avoidable. Invariants are state expressions that always hold. Invariants hold initially and before and after each transition. A “P-invariant,” or place-invariant, is a type of invariant constructed from place names such that transitions do not change the validity of the expression. Place-invariants can be represented as a vector of places which always sum to the same number of tokens. Other useful constructs include transition invariants and descending variants. A “T-invariant,” or transition-invariant, can be represented as a vector that indicates how often each transition has to fire to produce the same marking. A “descending variant” is a type of state expression that is true if for all transitions, the value of the state expression at the precondition state is greater than the value at the postcondition state in terms of a high-level Petri net’s algebra of objects and operations.

As instructive examples, the network “echo algorithm” is validated in Kindler, Reisig, Volzer, and Walter14 with Petri nets, and the “stop-and-wait” data link protocol is validated with colored Petri nets in Kristensen, Christensen, and Jensen.15

Evolving Algebras

Evolving algebras may be used to specify and validate a system at each level of abstraction.

An algebra provides a mathematical language for domains and functions. Evolving algebras build on the concept of finite state machines, using algebras of domains and functions as states in an abstract state
Appendix A. (Continued)

machine. Functions are dynamic in the sense that they may be paired with a tuple in a state for a location; states map the locations to their content. “Function updates” represent any sort of destructive assignment, where the mappings from locations to elements change. An evolving algebra is a set of transition rules where consequences consist of a finite number of function updates, including domain extensions or element deletions. When such a rule is applied to a static algebra of domains and functions, the consequence is another algebra differing only by the new values for the functions at arguments updated by the rule.\textsuperscript{16}

As an instructive example, the Kermit protocol is validated using evolving algebras in Huggins,\textsuperscript{17} where induction is used to prove protocol invariants manually.

Abstract state machines have additionally been used to verify hardware organization and architecture as well as procedural, functional, and object-oriented compilers.

A2.1.4 Logic

Some types of software, like knowledge bases, may exist in a declarative formal specification that is suitable for demonstrating consistency and completeness. An expert system may be thought of as consisting of a knowledge base, inference engine, and explanation unit. The knowledge base contains rules from experts, the inference engine uses this knowledge to reach conclusions given certain facts, and the explanation unit serves to explain how the conclusions were reached.

Inference

The validity of an inference procedure can be argued using a model-theoretic interpretation of rules as given by Ginsberg\textsuperscript{9} roughly as follows. In first-order logic, an inference procedure is sound if whenever a conclusion is inferred from a fact, the fact entails the conclusion. \textit{Modus ponens}, for example, is sound. This can be proven by contradiction by creating a model where \textit{modus ponens} holds but the conclusion does not. An inference procedure is complete if whenever a fact entails a conclusion, the conclusion can be inferred from the fact that every valid consequence can be found using the procedure. \textit{Modus ponens} is complete for databases specified in a form where an intersection of atomic conditions implies an atomic conclusion. Predicate logic can be normalized to a form consisting of intersections of atomic facts implying unions of conclusions. The inference procedure called resolution is both sound and complete for this normal form. Ginsberg\textsuperscript{9} provides a normalization procedure and related proofs.

Correctness

Given a set of conditions, inconsistent conclusions should not be reachable; a fact and its negation should not both hold given valid input. Domain experts should be consulted to ensure semantic consistency, that facts do not contradict in a way that is not directly expressed in logic. Completeness in an expert system ensures that given facts, some conclusion or goal may be reached.

Wentworth, Knaus, and Aougab\textsuperscript{18} provide a good guide for proof of correctness and demonstrate that knowledge bases can be partitioned, partitions nonrecursively proven correct, and finally recursively proven correct.

Partitioning

Some knowledge bases may be expressed in subsystems by design. In other cases, if goals can be grouped into known subsystems, working backward, intermediary variables and conditions that contribute to
Appendix A. (Continued)

reaching the goals for a subsystem can be included in its partition. If goals do not necessarily break into semantically apparent partitions, incidence matrices representing dependency from functional relationships between variables can be used. The composition of two relations is the product of their immediate dependency matrices, where nonzero elements of the product matrix are replaced with ones. The union of the dependencies is represented by matrix addition, where nonzero elements of the summation matrix are replaced with ones. The dependency relation for the knowledge base can be expressed by repeatedly applying successive multiplications of the immediate dependency matrix to sequentially higher degrees and summing across iterations until the matrix no longer changes. The dependency relation’s matrix columns represent groups of highly correlated variables.

Wentworth, Knaus, and Aougab introduce “Hoffman Regions” as maximal subsets of the input space of a knowledge base where each atomic formula has a single truth value. A unique set of these regions cover and partition the input space, and so in absence of rules that run external numerical software, a test case need only be run for each region. Two input variables that determine the same set of truth values for all atomic formulas in the knowledge base can be considered equivalent. The procedure given for producing the regions is to group and sort formulas containing a single variable so that each variable’s rules are grouped together and relational conditions sorted by the compared literal, then by the relational operator from less-than to greater-than. Formulas with multiple variables are assigned default regions TRUE and FALSE to avoid determining when combinations are true, numeric intervals are created for the truth value constraint, and the list of possible string values for each string variable is used. The Cartesian product of the regions for all variables comprises the Hoffmann regions of the knowledge base. If multiple variables are used in a formula, inconsistency demonstrated for the region containing the formula is an inconsistency of the expert system if the region is satisfiable.

Anomaly Detection

Anomaly detection is a validation method for which tools have been developed to help verify knowledge-based systems. Preece and Shinghal provided a formal foundation for anomaly detection as a validation method. In Preece and Shinghal, the four major types of anomaly are given as redundancy, ambivalence, circularity, and deficiency. Redundancy includes unsatisfiable conditions and redundant rules. Ambivalence represents inconsistency in terms of a semantic constraint violation, where all the mutually inconsistent conditions of a constraint hold for some instance obtained by a substitution.

Deficiency represents incompleteness where no output is produced for an environment, for example, in cases where rules are missing or some declared input is not used in conditions or goals.

Integrity checks and rule extension checks are two anomaly detection techniques. Integrity checks directly check for incompleteness in terms of unsatisfiable conditions, unusable consequents, and unused input. Rule checks can directly compare all rules in pairs to detect redundancy and ambivalence. Rule extension checks detect redundancy, ambivalence, circular dependency, and deficiency by computing every inference chain and applying the checks to the chains.

A3 Validating Tools

A variety of software packages are available for specification, simulation, state-space analysis, and automated theorem proving.

SPIN is a free software package from Bell Labs for simulation and state-space analysis for use in validating protocols using the PROMELA modeling language. See http://spinroot.com/spin/whatispin.html.
Appendix A. (Continued)

The Design/CPN tool is a free tool for design and analysis using colored Petri nets. See http://www.daimi.au.dk/designCPN/.

PVS is an environment for formal specification and theorem proving. See http://pvs.csl.sri.com.

HL7 is an application layer standard commonly used by laboratory information systems. The HL7 community provides tools and services that can assist in design, validation, and conformance testing.

- The HL7 Conformance Special Interest Group has developed a registry for a unique identification for XML validation for message profiles. Message Workbench is a free tool that can additionally assist analyzing and defining messaging requirements. For more information, see http://pvs.csl.sri.com under Special Interest Groups/Conformance/Documents and Presentations.

- To test and certify the messages, the Australian Healthcare Messaging Laboratory provides a free online testing tool to test messages against particular profiles. For more information, see http://www.ahml.com.au.

A4 Storage of Validation Data

Validation data include:

- input data related to system requirements or specific features;
- optional additional annotation regarding the testing procedure;
- data related to the execution of the test like the tester, date, and time;
- configuration data like product identifiers and version numbers for each component for the test; and
- output data produced from the input data for the execution using the configuration.

The combination of input data and configuration data should functionally produce the output data. To validate a particular fix, the input data specific to the test remains the same for both the control case of the original configuration and the test case of the new configuration, and the output for both can be stored. These tests can be considered related by the problem.

Document management systems, version control systems, problem tracking systems, relational databases, file system-based storage, and physical binders all provide means of managing this type of data.

Electronic storage is convenient and allows for an automated approach. Electronically storing input data facilitates testing. Electronically storing output data simplifies access for review. The volume of data is likely to be low and the data are not likely to be analyzed in summary, rather the primary requirement is to facilitate access to validation by configuration, and the related requirements, features, or problems for which the tests apply. The ability to track changes in configuration is additionally useful.

The lifetime of the system validation data should at least extend to the lifetime for process validation data. Given the relative infrequency of configuration changes, a purge policy or an option like optical storage for archiving data may not be necessary.

From a vendor’s perspective, a version control and configuration management system may prove useful for grouping output of test cases with a given configuration of software, and providing a way to store unit tests with the version of software tested. From a client’s perspective, if a procedure already exists for managing validation data of other types of products like devices, instruments, or information systems, consistency is advisable.
References for Appendix A


# Appendix B. Sample Change/Addition Request Form

<table>
<thead>
<tr>
<th>Name of facility and department</th>
<th>Requested by:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Phone: Date:</td>
</tr>
</tbody>
</table>

## CHANGE/ADDITION REQUEST

**To be completed by requestor:**

- **Module:**
  - Lab
  - Micro
  - BB
  - AP
  - Orders
  - Charts
  - Pharm
  - Rad
  - Registration

- **Bench or Testing Location:**
- **Description of Addition or Change:**

<table>
<thead>
<tr>
<th>Other Information</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Special Instructions:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
</tr>
</tbody>
</table>

**Requested completion date:**

**Other Information**
- Billing Required
- Instrument Interface/Autoverify
- ICD9 Code Required
- Expedite Print
- Point of Care
- Calculation

**Changes Performed by:**

**Notified:**

## VALIDATION

**To be completed by the requesting department. Validate all items/scenarios that apply to the Addition or Change. Attach validation script and print screens to this form.**

- Patient Registration
- Specimen Processing
- Order Entry
- Instrument Interface
- Data Entry
- High/Low/Abnormal Flags
- Reference Ranges/Panic Values
- Security/System Messages
- Calculation
- Reports/Charts
- Corrections
- Comments/Footnotes
- Operations
- Database
- Data Transmission to Other Systems
- Data Storage/Retrieval
- Other:

<table>
<thead>
<tr>
<th>Validation Performed by:</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Validation Approved by:  | Date: |
| (Department Director or Designee) |       |

<table>
<thead>
<tr>
<th>Validation Reviewed and Approved by:</th>
<th>Date:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Appendix C. Example of Change/Addition Request

<table>
<thead>
<tr>
<th>Name of facility and department</th>
<th>Requested by: Molly Tech</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Phone: x22341 Date: June 20, 2003</td>
</tr>
</tbody>
</table>

### CHANGE/ADDITION REQUEST

To be completed by requestor:

<table>
<thead>
<tr>
<th>Module:</th>
<th>Lab</th>
<th>Micro</th>
<th>BB</th>
<th>AP</th>
<th>Orders</th>
<th>Charts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>X</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Bench or Testing Location: | Routine Chemistry |

<table>
<thead>
<tr>
<th>Description of Addition or Change:</th>
<th>Change Glucose reference ranges for Age 0-1 month, male and female, to:</th>
</tr>
</thead>
<tbody>
<tr>
<td>Low – 42</td>
<td>Low Critical – 28</td>
</tr>
<tr>
<td>High – 89</td>
<td>High Critical – 200</td>
</tr>
</tbody>
</table>

| Special Instructions: |

<table>
<thead>
<tr>
<th>Other Information</th>
</tr>
</thead>
<tbody>
<tr>
<td>□ Billing Required</td>
</tr>
<tr>
<td>□ Instrument Interface/Autoverify</td>
</tr>
<tr>
<td>□ ICD9 Code Required</td>
</tr>
<tr>
<td>□ Expedite Print</td>
</tr>
<tr>
<td>□ Point of Care</td>
</tr>
<tr>
<td>□ Calculation</td>
</tr>
</tbody>
</table>

### CHANGES PERFORMED

<table>
<thead>
<tr>
<th>Changes Performed by:</th>
<th>LIS Tech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>June 24, 2003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Notified:</th>
<th>Molly Tech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>June 24, 2003</td>
</tr>
</tbody>
</table>

### VALIDATION

To be completed by the requesting department. Validate all items/scenarios that apply to the Addition or Change. Attach validation script and print screens to this form.

| □ Patient Registration |
| □ Specimen Processing |
| □ Order Entry |
| □ Instrument Interface |
| □ Data Entry | **X** High/Low/Abnormal Flags | **X** Reference Ranges/Panic Values |
| □ Security/System Messages |
| □ Calculation |
| □ Reports/Charts |
| □ Comments/Footnotes |
| □ Operations |
| □ Database |
| □ Data Transmission to Other Systems |

<table>
<thead>
<tr>
<th>Validation Performed by:</th>
<th>Molly Tech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>June 24, 2003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validation Approved by:</th>
<th>Dr. Director</th>
</tr>
</thead>
<tbody>
<tr>
<td>(Department Director or Designee)</td>
<td>Date: June 24, 2003</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Validation Reviewed and Approved by:</th>
<th>LIS Tech</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date:</td>
<td>June 24, 2003</td>
</tr>
</tbody>
</table>
Appendix D. Validation Script

1. Register two test patients: Patient A, male age 6 days, and Patient B, female age 2 weeks. Order five glucose tests, with difference collection times, on each patient.

2. Enter the following Glucose results. Enter appropriate footnote for all critical results.

<table>
<thead>
<tr>
<th>Patient A</th>
<th>Expected Results Flag</th>
<th>Patient B</th>
<th>Expected Results Flag</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>C</td>
<td>28</td>
<td>C</td>
</tr>
<tr>
<td>41</td>
<td>L</td>
<td>42</td>
<td>L</td>
</tr>
<tr>
<td>60</td>
<td>None</td>
<td>70</td>
<td>None</td>
</tr>
<tr>
<td>92</td>
<td>H</td>
<td>89</td>
<td>H</td>
</tr>
<tr>
<td>201</td>
<td>C</td>
<td>200</td>
<td>C</td>
</tr>
</tbody>
</table>

Provide print screens for items 3, 4, and 5. Print reports for item 5.

3. Did the expected result flags display at result entry with the appropriate visual (red font for critical, bold font for High/Low)?
   __ Yes __ No  If no, complete a corrective action form and submit to appropriate LIS staff.

4. Did the message “Critical Value. Call result” display at result entry for the appropriate critical values?
   __ Yes __ No  If no, complete a corrective action form and submit to appropriate LIS staff.

5. Did all result flags display correctly on the printed patient report?
   __ Yes __ No  If no, complete a corrective action form and submit to appropriate LIS staff.

6. Did the expected result flags transmit to the Data Repository system and display with the appropriate visual (red font for critical, bold font for High/Low)?
   __ Yes __ No  If no, complete a corrective action form and submit to appropriate LIS staff.

7. Was validation performed as expected? _______ Yes _____ No

8. Was additional validation required? _______ Yes _____ No
**The Quality System Approach**

Clinical and Laboratory Standards Institute (CLSI) subscribes to a quality system approach in the development of standards and guidelines, which facilitates project management; defines a document structure via a template; and provides a process to identify needed documents. The approach is based on the model presented in the most current version of CLSI/NCCLS HS1—*A Quality Management System Model for Health Care*. The quality system approach applies a core set of “quality system essentials” (QSEs), basic to any organization, to all operations in any healthcare service’s path of workflow (i.e., operational aspects that define how a particular product or service is provided). The QSEs provide the framework for delivery of any type of product or service, serving as a manager’s guide. The quality system essentials (QSEs) are:

- **Documents & Records**
- **Organization**
- **Personnel**
- **Equipment**
- **Purchasing & Inventory**
- **Process Control**
- **Information Management**
- **Occurrence Management**
- **Assessment**
- **Process Improvement**
- **Service & Satisfaction**
- **Facilities & Safety**

AUTO8-P addresses the quality system essentials (QSEs) indicated by an “X.” For a description of the other Clinical and Laboratory Standards Institute documents listed in the grid, please refer to the Related Clinical and Laboratory Standards Institute Publications section on the following page.

<table>
<thead>
<tr>
<th>Documents &amp; Records</th>
<th>Organization</th>
<th>Personnel</th>
<th>Equipment</th>
<th>Purchasing &amp; Inventory</th>
<th>Process Control</th>
<th>Information Management</th>
<th>Occurrence Management</th>
<th>Assessment</th>
<th>Process Improvement</th>
<th>Service &amp; Satisfaction</th>
<th>Facilities &amp; Safety</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>X AUTO3</td>
<td>AUTO4 LIS4 GP19</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>AUTO8-P</td>
<td></td>
<td></td>
<td>M29</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Adapted from CLSI/NCCLS document HS1—*A Quality Management System Model for Health Care*.

**Path of Workflow**

A path of workflow is the description of the necessary steps to deliver the particular product or service that the organization or entity provides. For example, CLSI/NCCLS document GP26—*Application of a Quality Management System Model for Laboratory Services* defines a clinical laboratory path of workflow which consists of three sequential processes: preanalytic, analytic, and postanalytic. All clinical laboratories follow these processes to deliver the laboratory’s services, namely quality laboratory information.

AUTO8-P addresses the clinical laboratory path of workflow steps indicated by an “X.” For a description of the other documents listed in the grid, please refer to the Related CLSI/NCCLS Publications section on the following page.

<table>
<thead>
<tr>
<th>Preanalytic</th>
<th>Analytic</th>
<th>Postanalytic</th>
</tr>
</thead>
<tbody>
<tr>
<td>Patient Assessment</td>
<td>Test Request</td>
<td>Specimen Collection</td>
</tr>
<tr>
<td>Specimen Transport</td>
<td>Specimen Receipt</td>
<td>Testing Review</td>
</tr>
<tr>
<td>Laboratory Interpretation</td>
<td>Results Report</td>
<td>Post-test Specimen Management</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Preanalytic</th>
<th>Analytic</th>
<th>Postanalytic</th>
</tr>
</thead>
<tbody>
<tr>
<td>X</td>
<td></td>
<td>X</td>
</tr>
</tbody>
</table>

Adapted from CLSI/NCCLS document HS1—*A Quality Management System Model for Health Care*. 

©Clinical and Laboratory Standards Institute. All rights reserved.
Related CLSI/NCCLS Publications

AUTO3-A Laboratory Automation: Communications with Automated Clinical Laboratory Systems, Instruments, Devices, and Information Systems; Approved Standard (2000). This document provides standards to facilitate accurate and timely electronic exchange of data and information between the automated laboratory elements.

AUTO4-A Laboratory Automation: Systems Operational Requirements, Characteristics, and Information Elements; Approved Standard (2001). This document describes operational requirements, characteristics, and required information elements of clinical laboratory automation systems. This information is used to determine the status of a clinical specimen within the clinical laboratory automation system, as well as the status of the actual components of the clinical laboratory automation system.

GP19-A2 Laboratory Instruments and Data Management Systems: Design of Software User Interfaces and End-User Software Systems Validation, Operation, and Monitoring; Approved Guideline—Second Edition (2003). This document identifies important factors that designers and laboratory managers should consider when developing new software-driven systems and selecting software user interfaces. Also included are simple rules to help prepare validation protocols for assessing the functionality and dependability of software.

LIS4-A Standard Guide for Documentation of Clinical Laboratory Computer Systems (2003). This guide covers documentation (defined as the information needed to install, use, maintain, or modify the system) for a computer system operating in a clinical laboratory.

M29-A2 Protection of Laboratory Workers from Occupationally Acquired Infections; Approved Guideline—Second Edition (2001). Based on U.S. regulations, this document provides guidance on the risk of transmission of hepatitis viruses and human immunodeficiency viruses in any laboratory setting; specific precautions for preventing the laboratory transmission of blood-borne infection from laboratory instruments and materials; and recommendations for the management of blood-borne exposure.

* Proposed-level documents are being advanced through the Clinical and Laboratory Standards Institute consensus process; therefore, readers should refer to the most recent editions.