

Technical Information Report

AAMI TIR36:2007

Validation of software for regulated processes



Association for the Advancement
of Medical Instrumentation



Documents

Document Sharing Hub

Validation of software for regulated processes

Developed by
Association for the Advancement of Medical Instrumentation

Approved 13 December 2007 by
Association for the Advancement of Medical Instrumentation

Abstract: Applies to any software used to automate device design, testing, component acceptance, manufacturing, labeling, packaging, distribution, and complaint handling or to automate any other aspect of the quality system as defined by the Quality System Regulation (21 CFR 820). In addition, it applies to software used to create, modify, and maintain electronic records and to manage electronic signatures that are subject to the validation requirements (21 CFR 11). This TIR can also be broadly applied wherever software automates processes regulated by the FDA. This TIR applies to software used in the production of a device and to software used in implementation of the device manufacturer's quality system. It does not apply to software used as a component, part, or accessory of a medical device or software that is itself a medical device.

Keywords: medical device software, medical electrical equipment, electromedical equipment, risk management

AAMI Technical Information Report

A technical information report (TIR) is a publication of the Association for the Advancement of Medical Instrumentation (AAMI) Standards Board that addresses a particular aspect of medical technology.

Although the material presented in a TIR may need further evaluation by experts, releasing the information is valuable because the industry and the professions have an immediate need for it.

A TIR differs markedly from a standard or recommended practice, and readers should understand the differences between these documents.

Standards and recommended practices are subject to a formal process of committee approval, public review, and resolution of all comments. This process of consensus is supervised by the AAMI Standards Board and, in the case of American National Standards, the American National Standards Institute.

A TIR is not subject to the same formal approval process as a standard. However, a TIR is approved for distribution by a technical committee and the AAMI Standards Board.

Another difference is that, although both standards and TIRs are periodically reviewed, a standard must be acted on—reaffirmed, revised, or withdrawn—and the action formally approved usually every 5 years but at least every 10 years. For a TIR, AAMI consults with a technical committee about 5 years after the publication date (and periodically thereafter) for guidance on whether the document is still useful—that is, to check that the information is relevant or of historical value. If the information is not useful, the TIR is removed from circulation.

A TIR may be developed because it is more responsive to underlying safety or performance issues than a standard or recommended practice or because achieving consensus is extremely difficult or unlikely. Unlike a standard, a TIR permits the inclusion of differing viewpoints on technical issues.

CAUTION NOTICE: This AAMI TIR may be revised or withdrawn at any time. Because it addresses a rapidly evolving field or technology, readers are cautioned to ensure that they have also considered information that may be more recent than this document.

All standards, recommended practices, technical information reports, and other types of technical documents developed by AAMI are *voluntary*, and their application is solely within the discretion and professional judgment of the user of the document. Occasionally, voluntary technical documents are adopted by government regulatory agencies or procurement authorities, in which case the adopting agency is responsible for enforcement of its rules and regulations.

Comments on this technical information report are invited and should be sent to AAMI, Attn: Standards Department, 1110 N. Glebe Road, Suite 220, Arlington, VA 22201-4795.

Published by
Association for the Advancement of Medical Instrumentation
1110 N. Glebe Road, Suite 220
Arlington, VA 22201-4795
www.aami.org

© 2008 by the Association for the Advancement of Medical Instrumentation

All Rights Reserved

Publication, reproduction, photocopying, storage, or transmission, electronically or otherwise, of all or any part of this document without the prior written permission of the Association for the Advancement of Medical Instrumentation is strictly prohibited by law. It is illegal under federal law (17 U.S.C. § 101, *et seq.*) to make copies of all or any part of this document (whether internally or externally) without the prior written permission of the Association for the Advancement of Medical Instrumentation. Violators risk legal action, including civil and criminal penalties, and damages of \$100,000 per offense. For permission regarding the use of all or any part of this document, contact AAMI at 1110 N. Glebe Road, Suite 220, Arlington, VA 22201-4795. Phone: (703) 525-4890; Fax: (703) 525-1067.

Printed in the United States of America

ISBN 1-57020-306-7

Contents

	Page
Glossary of equivalent standards.....	v
Committee representation.....	vi
Foreword	viii
Introduction	x
1 General	1
1.1 Purpose and intent.....	1
1.2 Scope.....	1
1.3 Document organization.....	2
2 Regulatory context.....	2
2.1 Context of 21 CFR 820.70(i), Automated processes.....	2
2.2 Context of the Quality System Regulation (QSR) – 21 CFR 820.....	3
2.3 Context of 21 CFR 11.....	3
2.4 Context of the <i>General Principles of Software Validation</i>	4
3 Software validation discussion.....	4
3.1 Definition.....	4
3.2 Confidence-building activities: The tools in the toolbox.....	4
3.3 Critical thinking	5
4 Software validation and critical thinking.....	5
4.1 Overview.....	5
4.2 In scope?	9
4.3 <i>Develop</i> phase.....	10
4.4 <i>Maintain</i> phase	19
4.5 <i>Retire</i> phase	20
5 Documentation.....	21
6 Prerequisite processes	21
Annexes	
A The toolbox	23
B Risk management.....	38
C Examples	46
Example 1: PLC for manufacturing equipment.....	47
Example 2: Automated welding system.....	50
Example 3: Automated welding process control system	52
Example 4: C/C++ language compiler	58
Example 5: Automated Software Test System	62
Example 6: A simple spreadsheet	66
Example 7: A (not so) simple spreadsheet.....	69
Example 8: Parametric sterilizer	73
Example 9: Nonconforming material reporting system—Total system upgrade	77
Example 10: Software for scheduling nonconforming material report review board meetings	81
Example 11: Approved vendor list system.....	83
Example 12: Calibration management software	87
Example 13: Automated vision system.....	91
Example 14: Pick and place system	93
D Definitions.....	96
E Bibliography	99

Glossary of equivalent standards

International Standards adopted in the United States may include normative references to other International Standards. For each International Standard that has been adopted by AAMI (and ANSI), the table below gives the corresponding U.S. designation and level of equivalency to the International Standard. NOTE: Documents are sorted by international designation.

Other normatively referenced International Standards may be under consideration for U.S. adoption by AAMI; therefore, this list should not be considered exhaustive.

International designation	U.S. designation	Equivalency
IEC 60601-1:2005	ANSI/AAMI ES60601-1:2005	Major technical variations
IEC 60601-1-2:2007	ANSI/AAMI/IEC 60601-1-2:2007	Identical
IEC 60601-2-2:2006	ANSI/AAMI/IEC 60601-2-2:2006	Identical
IEC 60601-2-4:2002	ANSI/AAMI DF80:2003	Major technical variations
IEC 60601-2-19:1990 and Amendment 1:1996	ANSI/AAMI I136:2004	Major technical variations
IEC 60601-2-20:1990 and Amendment 1:1996	ANSI/AAMI I151:2004	Major technical variations
IEC 60601-2-21:1994 and Amendment 1:1996	ANSI/AAMI/IEC 60601-2-21 and Amendment 1:2000 (consolidated texts)	Identical
IEC 60601-2-24:1998	ANSI/AAMI ID26:2004	Major technical variations
IEC 60601-2-47:2001	ANSI/AAMI EC38:2007	Major technical variations
IEC 60601-2-50:2001	ANSI/AAMI/IEC 60601-2-50:2006	Identical
IEC/TR 60878:2003	ANSI/AAMI/IEC TIR60878:2003	Identical
IEC/TR 62296:2003	ANSI/AAMI/IEC TIR62296:2003	Identical
IEC 62304:2006	ANSI/AAMI/IEC 62304:2006	Identical
IEC/TR 62348:2006	ANSI/AAMI/IEC TIR62348:2006	Identical
ISO 5840:2005	ANSI/AAMI/ISO 5840:2005	Identical
ISO 7198:1998	ANSI/AAMI/ISO 7198:1998/2001/(R)2004	Identical
ISO 7199:1996	ANSI/AAMI/ISO 7199:1996/(R)2002	Identical
ISO 8637:2004	ANSI/AAMI RD16:2007	Major technical variations
ISO 8638:2004	ANSI/AAMI RD17:2007	Major technical variations
ISO 10993-1:2003	ANSI/AAMI/ISO 10993-1:2003	Identical
ISO 10993-2:2006	ANSI/AAMI/ISO 10993-2:2006	Identical
ISO 10993-3:2003	ANSI/AAMI/ISO 10993-3:2003	Identical
ISO 10993-4:2002 and Amendment 1:2006	ANSI/AAMI/ISO 10993-4:2002 and Amendment 1:2006	Identical
ISO 10993-5:1999	ANSI/AAMI/ISO 10993-5:1999	Identical
ISO 10993-6:2007	ANSI/AAMI/ISO 10993-6:2007	Identical
ISO 10993-7:1995	ANSI/AAMI/ISO 10993-7:1995/(R)2001	Identical
ISO 10993-9:1999	ANSI/AAMI/ISO 10993-9:1999/(R)2005	Identical
ISO 10993-10:2002 and Amendment 1:2006	ANSI/AAMI BE78:2002 ANSI/AAMI BE78:2002/A1:2006	Minor technical variations Identical
ISO 10993-11:2006	ANSI/AAMI/ISO 10993-11:2006	Identical
ISO 10993-12:2007	ANSI/AAMI/ISO 10993-12:2007	Identical
ISO 10993-13:1998	ANSI/AAMI/ISO 10993-13:1999/(R)2004	Identical
ISO 10993-14:2001	ANSI/AAMI/ISO 10993-14:2001/(R)2006	Identical
ISO 10993-15:2000	ANSI/AAMI/ISO 10993-15:2000/(R)2006	Identical
ISO 10993-16:1997	ANSI/AAMI/ISO 10993-16:1997/(R)2003	Identical
ISO 10993-17:2002	ANSI/AAMI/ISO 10993-17:2002	Identical

International designation	U.S. designation	Equivalency
ISO 10993-18:2005	ANSI/AAMI BE83:2006	Major technical variations
ISO/TS 10993-19:2006	ANSI/AAMI/ISO TIR10993-19:2006	Identical
ISO/TS 10993-20:2006	ANSI/AAMI/ISO TIR10993-20:2006	Identical
ISO 11135-1:2007	ANSI/AAMI/ISO 11135-1:2007	Identical
ISO 11137-1:2006	ANSI/AAMI/ISO 11137-1:2006	Identical
ISO 11137-2:2006 (2006-08-01 corrected version)	ANSI/AAMI/ISO 11137-2:2006	Identical
ISO 11137-3:2006	ANSI/AAMI/ISO 11137-3:2006	Identical
ISO 11138-1: 2006	ANSI/AAMI/ISO 11138-1:2006	Identical
ISO 11138-2: 2006	ANSI/AAMI/ISO 11138-2:2006	Identical
ISO 11138-3: 2006	ANSI/AAMI/ISO 11138-3:2006	Identical
ISO 11138-4: 2006	ANSI/AAMI/ISO 11138-4:2006	Identical
ISO 11138-5: 2006	ANSI/AAMI/ISO 11138-5:2006	Identical
ISO/TS 11139:2006	ANSI/AAMI/ISO 11139:2006	Identical
ISO 11140-1:2005	ANSI/AAMI/ISO 11140-1:2005	Identical
ISO 11140-3:2007	ANSI/AAMI/ISO 11140-3:2007	Identical
ISO 11140-4:2007	ANSI/AAMI/ISO 11140-4:2007	Identical
ISO 11140-5:2007	ANSI/AAMI/ISO 11140-5:2007	Identical
ISO 11607-1:2006	ANSI/AAMI/ISO 11607-1:2006	Identical
ISO 11607-2:2006	ANSI/AAMI/ISO 11607-2:2006	Identical
ISO 11737-1: 2006	ANSI/AAMI/ISO 11737-1:2006	Identical
ISO 11737-2:1998	ANSI/AAMI/ISO 11737-2:1998	Identical
ISO 11737-3:2004	ANSI/AAMI/ISO 11737-3:2004	Identical
ISO 13485:2003	ANSI/AAMI/ISO 13485:2003	Identical
ISO 14155-1:2003	ANSI/AAMI/ISO 14155-1:2003	Identical
ISO 14155-2:2003	ANSI/AAMI/ISO 14155-2:2003	Identical
ISO 14160:1998	ANSI/AAMI/ISO 14160:1998	Identical
ISO 14161:2000	ANSI/AAMI/ISO 14161:2000	Identical
ISO 14937:2000	ANSI/AAMI/ISO 14937:2000	Identical
ISO/TR 14969:2004	ANSI/AAMI/ISO TIR14969:2004	Identical
ISO 14971:2007	ANSI/AAMI/ISO 14971:2007	Identical
ISO 15223-1:2007	ANSI/AAMI/ISO 15223-1:2007	Identical
ISO 15225:2000 and A1:2004	ANSI/AAMI/ISO 15225:2000/(R)2006 and A1:2004/(R)2006	Identical
ISO 15674:2001	ANSI/AAMI/ISO 15674:2001	Identical
ISO 15675:2001	ANSI/AAMI/ISO 15675:2001	Identical
ISO 15882:2003	ANSI/AAMI/ISO 15882:2003	Identical
ISO/TR 16142:2006	ANSI/AAMI/ISO TIR16142:2005	Identical
ISO 17664:2004	ANSI/AAMI ST81:2004	Major technical variations
ISO 17665-1:2006	ANSI/AAMI/ISO 17665-1:2006	Identical
ISO 18472:2006	ANSI/AAMI/ISO 18472:2006	Identical
ISO/TS 19218:2005	ANSI/AAMI/ISO 19218:2005	Identical
ISO 22442-1:2007	ANSI/AAMI/ISO 22442-1:2007	Identical
ISO 22442-2:2007	ANSI/AAMI/ISO 22442-2:2007	Identical
ISO 22442-3:2007	ANSI/AAMI/ISO 22442-3:2007	Identical
ISO 25539-1:2003 and A1:2005	ANSI/AAMI/ISO 25539-1:2003 and A1:2005	Identical

Committee representation

Association for the Advancement of Medical Instrumentation

AAMI Medical Device Software Committee

This technical information report (TIR) was developed by the AAMI Medical Device Software Committee. Committee approval of the TIR does not necessarily imply that all committee members voted for its approval.

At the time this document was published, the **AAMI Medical Device Software Committee** had the following members:

<i>Chairs:</i>	Sherman Eagles John F. Murray Jr.
<i>Members:</i>	Randy Armstrong, Cyberonics Inc. David R. Christie, Spacelabs Medical Inc. Theresa Dennis, Sterigenics International Andrew Dunham, Baxter Healthcare Corp. Sherman Eagles, Medtronic Inc. Christine Flahive, Belle Mead, NJ Larry Fry, Draeger Medical Nancy George, Towson, MD Ron Gerner, Abbott Laboratories Steven Gitelis, GB Lumina Inc. Lori Haller, Steris Corp. James Hempel, Covidien Sam Jarrell, CerTech LLC Jeremy Jensen, Guidant/Boston Scientific Corp. David R. Jones, Philips Medical Systems Martin J. King, Hospira Worldwide Inc. Alan Kusnitz, SoftwareCPR Bernie Liebler, Advanced Medical Technology Association (AdvaMed) Don Lin, Irvine, CA Steve Mallory, Welch Allyn Inc. Mark Maritch, Datascope Corp. Don McAndrews, Respironics Inc. Mary Beth McDonald, St. Jude Medical Dennis Mertz, Becton Dickinson John F. Murray Jr., U.S. Food and Drug Administration Raj Raghavendran, Johnson & Johnson/Ethicon Endo-Surgery Bill Riley, Hill-Rom Company Harvey Rudolph, Underwriters Laboratories Inc. Richard Schrenker, Massachusetts General Hospital Xianyu Shea, Stryker Medical Division Carla Sivak, Edwards Lifesciences Scott Thiel, Roche Diagnostics Corp. Ann Vu, Bausch & Lomb Inc. James Webb, Cardinal Health Andrew Whitman, National Electrical Manufacturers Association Gregory Whitney, CR Bard
<i>Alternates:</i>	Aziz Bhai, Hill-Rom Company Christopher P. Clark, Bausch & Lomb Inc. Rich Eaton, National Electrical Manufacturers Association Christopher Ganser, CR Bard Jeff Gilham, Spacelabs Medical Inc. Steve Hellstrom, Hospira Worldwide Inc. Denise Stearns Holliman, Boston Scientific Corp. Gene Kelly, CerTech LLC Patricia Krantz, Medtronic Inc. Gretel Lumley, Philips Medical Systems David Michel, Steris Corp.

Dewey Phan, Becton Dickinson
Rodney Rasmussen, Abbott Laboratories
Miguel Rodriguez, Johnson & Johnson/Cordis
Robert Smith, St. Jude Medical
Donna-Bea Tillman, U.S. Food and Drug Administration

Acknowledgments

The AAMI Medical Device Software Committee gratefully acknowledges the work of its Validation of Software for Regulated Processes Task Group, which handled the development of this TIR. The Task Group has the following members:

Chairs: Denise Stearns (Holliman)
Steve Gitelis
Members: Mark Allen, Bonfils
Barbara Beiersdorf, Medtronic Inc.
Paul Brown, Medtronic Inc.
Steve Gitelis, GB Lumina Inc.
Denise Stearns (Holliman), Still River Systems, Inc.
Rich Hall, Eli Lilly
Jeremy Jensen, Guidant/Boston Scientific Corp.
Lisa Last, NxStage Medical, Inc.
John F. Murray Jr., U.S. Food and Drug Administration
Frank Scavo, Strativa
David Vogel, Intertech Engineering and Associates, Inc.
Carl Wyrwa, Beckman Coulter, Inc.

A special acknowledgment goes to the following contributors to this document: Debbie Iampietro, Jennifer V. Anderson, and Kathleen O'Donnell.

NOTE—Participation by federal agency representatives in the development of this document does not constitute endorsement by the federal government or any of its agencies.

Foreword

In the historical approach to validation, the terms *value added* and *software validation* tended to be mutually exclusive. Software validation historically may or may not have been value added. How well have the historical approaches truly ensured software performance according to its intended use? How often has “validated” software been deployed and still not performed as the users thought it should?

It is important to get the most value out of your software validation activities. After all, you or your company is committing valuable resources to the validation efforts; therefore, it is essential that you receive an appropriate return on this investment.

So, why do some people feel that they are not getting as much value out of their software validation activities as they should? Why do some people feel as if they have to do too much to achieve compliance with this requirement? Why do some people feel as if their software validation activities are not aligned with their business goals or interest? Why do some people feel as if their internal software validation activities are redundant when they use high-quality off-the-shelf (OTS) software? Why are some people doing too little or doing nothing at all? Why is there uncertainty about which software requires validation and which software does not?

This technical information report (TIR) is intended to help you understand the issues behind these questions and to give you suggestions on how to develop a more value-added approach to software validation.

It is important to note that a medical device regulation requiring software validation does exist. The regulation section, 21 CFR 820.70(i), Automated processes, is written in broad terms so that it can apply to all medical device manufacturers. The regulation section identifies a problem to be solved or an end point to be achieved, but it does not provide any information about how to solve this problem and meet the intent of this regulatory requirement. Other specific information that the U.S. Food and Drug Administration (FDA) has provided on this topic is contained in the *General Principles of Software Validation; Final Guidance for Industry and FDA Staff* (GPSV). Section 6 of the GPSV provides guidance on the validation of automated process equipment and quality system software.

This report is not establishing a new direction, but it represents a view of the issues from a medical device industry perspective and a description of methods many in the industry may already follow. This report is meant to be a step toward better understanding the industry’s perspective on how to be compliant with the regulation in a value-added way.

Over time, many practices have evolved into a checklist-mentality approach based on a compliance need. At times the checklist approach inadvertently causes activities to stray from value-added activities that appropriately substantiate that the software performs as intended. Straying occurs when a single solution is sought that is intended to satisfy a large number of stakeholders, each with a potentially different set of objectives and requirements. The stakeholders represent many perspectives focused on quality system implementation, regulatory needs, engineering practices, auditing and assessment requirements, business and legal needs, consulting services, and the like.

One of the key challenges is to find a solution that is aligned with the needs of all stakeholders, especially the needs of the individuals performing the validations and the auditors measuring the adequacy of the validations. The belief is that the manufacturer is expected to apply due diligence in the form of best practices in the areas of risk management, quality, and engineering, to create a solution that not only would satisfy this regulatory requirement but also would align with the intent of the regulation.

This report is intended to provide an awareness of concepts and tools that can be applied to the task of software validation. To begin, a simple analogy conveys the basic concept behind this report. A carpenter’s toolbox contains various tools such as hammers, wrenches, screwdrivers, and drills. When a carpenter is faced with a task, he or she chooses an appropriate tool that will complete the task in a safe and effective manner. For example, when a carpenter is nailing boards together, the most appropriate tool would be a hammer rather than a wrench or screwdriver. In addition, it is important to choose the type of hammer applicable to the user’s circumstances. A sledgehammer may get the job done, but it will probably leave the boards damaged or leave the user exhausted if there are a significant number of boards to be nailed together. However, if the sledgehammer is the only tool in the toolbox, the carpenter’s only choice is to use the inappropriate tool.

The sledgehammer analogy represents the one-size-fits-all type of validation that uses one set of tools for all regulated process software and is an example of not applying critical thinking. Like the sledgehammer, the one-size-fits-all type of validation gets the job done but at a price that does not always include value-added activities. In addition, it is probable that unidentified risks have not been properly controlled. In other words, a one-size-fits-all

checklist mentality typically creates extra work for simple, low-risk software or falls short of the work required for complex, high-risk software.

Software that needs to be validated might have many different intended uses and be used in many different scenarios, which involve very different risks. Different tools and associated approaches are necessary to accomplish an optimal validation for a vast variety of situations.

This report offers suggestions on how to apply critical thinking to determine the best approach for validation of software through selection of the best tools from the toolbox, thereby allowing implementation of value-added solutions that are both compliant and consistent with business requirements.

For the software validation efforts to be viewed as highly successful, the following statements should be true:

- The automated process or associated software functions as intended, without compromising device safety, device quality, or the integrity of the quality system.
- The people performing the necessary activities believe that the efforts are meaningful and worthwhile (i.e., least burdensome or most valuable activities).
- The manufacturer is in a state of compliance.
- Auditors and inspectors view the activities applied and the resulting records as acceptable evidence of compliance.

Introduction

This technical information report (TIR) has been developed to assist readers in determining appropriate activities for the validation of regulated process software using a risk-based approach that applies critical thinking.

This TIR is the result of an effort to bring together experience from medical device industry personnel who deal with performing this type of software validation and who are tasked with establishing auditable documentation. The TIR has been developed with certain questions and problems in mind that we all go through when faced with validating regulated process software, such as the following: What has to be done? How much is enough? How is risk analysis involved? After much discussion, the AAMI Validation of Software for Regulated Processes Task Group concluded that in every case a set of activities (i.e., the tools from the toolbox) was identified to provide a level of confidence in the ability of the software to perform according to its intended use. However, the list of activities varied depending on factors including, among others, the complexity of the software, the risk of harm involved, and the pedigree (e.g., quality, stability) of vendor-supplied software.

The resulting report includes two key elements:

- A method of applying critical thinking to identifying what needs to be completed for regulated process software validation. That method includes a risk-based approach that considers whether a software failure can cause harm.
- A toolbox of tools that can be used to establish a sufficient level of confidence that the software will perform as intended. It should be noted that such tools have been included on the basis of experience showing what works and what has not worked. The toolbox represents current knowledge of good software engineering practice. As more experience is gained and technology evolves, what works will also evolve, and the content of the toolbox may change.

NOTE—This introduction does not contain provisions of the AAMI TIR, *Validation of software for regulated processes* (AAMI TIR36:2007), but it does provide important information about the development and intended use of the document.

Validation of software for regulated processes

1 General

1.1 Purpose and intent

The purpose of this technical information report (TIR) is to provide guidance on what to think about when determining the appropriate content and size of a validation effort applied to software used for regulated processes. In addition, this TIR provides guidance on a method of reaching the appropriate depth and rigor of activities through analyzing and evaluating various aspects of the software and its environment. The TIR is intended to define this method through description, definition, and examples of applying critical thinking in a variety of circumstances.

This TIR is *not* intended to create a new U.S. Food and Drug Administration (FDA)–authorized minimum set of validation tasks and documentation. An Association for the Advancement of Medical Instrumentation (AAMI) TIR is a collection of “best practices,” and this document reflects the collective judgment of the AAMI Validation of Software for Regulated Processes Task Group of the best practices for validating software for regulated processes.

1.2 Scope

This TIR applies to any software used to automate device design, testing, component acceptance, manufacturing, labeling, packaging, distribution, and complaint handling or to automate any other aspect of the quality system, as defined by the Quality System Regulation (21 CFR 820), or QSR. In addition, the TIR applies to software used to create, modify, and maintain electronic records and to manage electronic signatures that are subject to the validation requirements (21 CFR 11). This TIR can also be broadly applied wherever software automates processes regulated by the FDA.

This TIR applies to

- software used in the production of a device, and
- software used in implementation of the device manufacturer’s quality system.

It does not apply to

- software used as a component, part, or accessory of a medical device, or
- software that is itself a medical device.

This TIR may provide useful information and recommendations to

- people responsible for determining the appropriate content and size of a validation effort;
- people responsible for performing the analyses and evaluations that drive the content or size determination;
- people responsible for planning and executing the validation activities;
- people responsible for reviewing and approving the adequacy of the validation effort; and
- people responsible for auditing, inspecting, and evaluating the validation for compliance to regulation.

The TIR discusses how the general provisions of the QSR apply to regulated process software and describes an approach to evaluating this software. However, the TIR does not list the tasks and activities that must be used to comply with the law. The TIR does not create or confer any rights for or on any person and does not operate to bind the user. An alternative approach may be used if such an approach satisfies the requirements of the applicable statute, regulations, or both. No specific methodology or specific validation technique or method is required or suggested by this TIR. For each software project, the responsible party should determine and justify the specific approach, the combination of software risk management activities to be used, and the level of effort to be applied. Specific training or experience in medical device quality management systems and the regulations governing these systems is recommended.

1.3 Document organization

This document is arranged as a main body with five annexes. The main body encompasses 21 pages from page 2 through page 22 and establishes the following:

- the context for validation of software for regulated processes;
- the concept of critical thinking and its relation to software validation;
- the application of critical thinking within the software life cycle, using a simplified waterfall process model as an example; and
- the systems and processes needed to support critical thinking and the software life cycle.

The five annexes contain the following information:

- Annex A is referred to throughout the document as the toolbox and contains more information about the various tools or confidence-building activities.
- Annex B is a brief discussion of risk management, including an example risk model.
- Annex C presents example studies demonstrating how critical thinking can be applied to software validation of regulated software in a variety of situations, including different complexities, pedigrees, and risk levels.
- Annexes D and E contain definitions and resources.

2 Regulatory context

The information in this section is meant to put the TIR in context with the current regulations and guidance documents. This section is meant to position the information in the TIR relative to the references below and represents the AAMI Validation of Software for Regulated Processes Task Group current understanding or interpretation of this information.

Several regulatory elements establish the complete context for the validation of regulated process software. These include:

- the specific section of the QSR, 21 CFR 820.70(i), Automated processes;
- the general concepts of quality systems as defined in the QSR (21 CFR 820);
- the specific guidance of 21 CFR 11; and
- the general guidance of the FDA's software guidance document, *General Principles of Software Validation*, (GPSV) including the specific guidance of section 6, "Validation of Automated Process Equipment and Quality System Software."

The sections that follow address the context of each of the regulatory elements.

2.1 Context of 21 CFR 820.70(i), Automated processes

Section 21 CFR 820.70(i) states, "When computers or automated data processing systems are used as part of production or the quality system, the manufacturer shall validate computer software for its intended use according to an established protocol. All software changes shall be validated before approval and issuance. These validation activities and results shall be documented."

When considering this context, we must carefully think through the following questions:

- Why does this regulation exist?
- What is the intent of this regulation?
- What problem are we really trying to solve?
- Why should we concern ourselves with software used to automate processes?
- Why are we required to validate this type of software?

It is the AAMI Medical Device Software Committee's understanding that the regulation exists to establish a level of confidence that the software will not adversely affect the safety of products or the quality of products. An adverse impact as a result of software failure could include the release of a defective product or the improper trending of complaint data such that the trending would not identify the need to perform a field action on a defective product. The committee also believes that the regulation exists to remind us to consider the risk of harm when bringing software into a process, when making changes to existing software in a process, or when simply adding the use of off-the-shelf (OTS) software. When software is brought in to automate a part of a process, it becomes an integral part of the process. All parts of a process, including people, tools, software, materials, and so forth, have the potential of adding an element of risk to the process and should be fully considered when assessing the risk of harm.

The stakeholders need to fully understand what they are entering into when they decide to automate a process or part of a process by using software. The whole idea is to find the most effective way to gain the necessary level of confidence in the software and the process it automates.

The committee's understanding of the intent of the regulation is to consider

- how the software fits into the overall process,
- what the software does,
- how we can determine that the software is working correctly,
- what potential risks are involved with the process that is automated by the software, and
- how we are going to manage these risks to an acceptable level.

2.1.1 Explanation of 21 CFR 820.70(i), Automated processes

An explanation to help understand the regulation requirements and the wording of the regulation is provided in this section, with some discussion regarding the committee's understanding of the intent of each section:

“automated data processing systems”: This terminology refers to a process that has been historically implemented through a manual, paper-driven method but is now automated through the use of computer software. Automated data processing systems can be confused with manufacturing process automation. The systems are actually a superset of manufacturing process automation and also include processes that implement the quality system activities.

“used as part of production or the quality system”: This phrase refers to software used in systems as diverse as manufacturing floor systems, corrective and preventive action (CAPA) systems, documentation systems, complaint-handling systems, product traceability systems, and preferred vendor systems,, among others.

“shall validate computer software”: This phrase means the *full* range of tasks necessary to establish confidence in the software. As defined in the regulatory context elements and in this TIR, it is *not* limited to testing.

“for its intended use”: This phrase refers to a formal statement of requirements and a clear definition of the process being automated and requires an understanding of what will be done with the software.

“according to an established protocol”: Although the word *protocol* is used with different meaning throughout the medical industry, here the word implies a plan. The plan for validation must be a formal, approved document or documents.

“All software changes shall be validated”: This phrase means that it is not sufficient to see validation as a one-time occurrence. Validation is a life-cycle activity that continues as long as the software is in use.

“These validation activities and results shall be documented”: This phrase means that the results of the validations are reasonably easily audited.

2.2 Context of the Quality System Regulation (QSR)—21 CFR 820

All formal quality systems such as ISO 13485, ISO 9000, and 21 CFR 820 have an underlying assumption that each quality system process must meet its intended purpose. For example, it is expected that a customer complaint management process meets all requirements established by the quality system. Internal audits are performed to ensure that a process is meeting these requirements according to the intended purpose of the process. Manufacturing processes have an explicit requirement to be validated before use in the manufacture of product. However, the QSR does not explicitly require that quality systems processes be validated before deployment. Therefore, confusion arises when a quality system process is automated by software. Is just the software being validated, or is the process with the software being validated? The implication in the QSR is not only that the software works, but also that the quality process itself functions in a way that meets its regulatory or corporate obligation.

2.3 Context of 21 CFR 11

Sections 21 CFR 11.10 and 21 CFR 11.10(a) state, in part, that “Persons who use ... systems to create, modify, maintain, or transmit electronic records shall employ procedures and controls designed to ensure the authenticity, integrity, and, when appropriate, the confidentiality of electronic records. Such procedures and controls shall include

...Validation of systems to ensure accuracy, reliability, consistent intended performance, and the ability to discern invalid or altered records.”

The validation requirement in this regulation is redundant to the explicit validation requirement defined in the medical device predicate rule 21 CFR 820.70(i). In addition, the QSR includes requirements that quality records be accurate and valid and that authorized individuals be accountable for performing regulated activities. The requirements carry over to the software in the event that the regulated process software incorporates electronic records, electronic signatures, or both, as defined by the regulation. Because the content of this TIR addresses this predicate rule, by default the TIR addresses the validation requirement in 21 CFR 11. The exact designs for ensuring accurate and valid electronic records and the execution of electronic signatures are not part of this TIR.

2.4 Context of the *General Principles of Software Validation*

It is the AAMI Validation of Software for Regulated Processes Task Group's perspective that the GPSV provides a good foundation for software validation from the perspective of software engineering and risk management. This TIR builds on that foundation with practical guidance for validation of software from the perspective of the quality system, looking at validation activities as a form of due diligence.

One of many essential elements of the GPSV is the concept that the risk of the software (in relation to medical device safety or efficacy) drives the rigor of the validation effort and that confidence-building activities are to be completed during the development and qualification of software. Such concepts are addressed frequently throughout the GPSV document. GPSV sections 1 through 5 are meant to be applied to software found in a medical device or to software that is a medical device. GPSV section 6 addresses software used for automated processes (the same topic as this TIR). Section 6 speaks to the need to apply the same level of life-cycle control and risk management concepts as defined in GPSV sections 1 through 5 on the basis of an assessment of what is appropriate for regulated process software. The content of this TIR defines a method to assist in determining what is appropriate. It is important to note that the GPSV stresses the importance of the basic principles of software validation (as shown in GPSV sections 1 through 5) in the development and implementation of software as part of production or the quality system. The GPSV defines *software validation* and is the basis of the software validation definition discussion in the next section of this TIR.

3 Software validation discussion

3.1 Definition

In FDA regulations and related guidance, the term software validation is used broadly to include all of the activities that lead to the conclusion that a piece of software is fit for its intended use. For example, page 6 of the *General Principles of Software Validation* says,

A conclusion that software is validated is highly dependent upon comprehensive software testing, inspections, analyses, and *other verification tasks* performed at each stage of the software development life cycle.

The term *software validation* has been interpreted both broadly and narrowly, from just testing to extensive activities including testing. This TIR uses the term *software validation* to denote all of the activities that establish a level of confidence that the software is appropriate for its intended use and that it is trustworthy and reliable. The chosen activities, whatever they may be, should ensure that the software works.

3.2 Confidence-building activities: The tools in the toolbox

The tools in the software validation toolbox include activities completed during the life cycle of software that reduce risk and build confidence (i.e., verification activities that support the conclusion that the software is validated). The tools are essentially a group of activities that have been established as value-added, risk control measures through empirical use over the past 30 years or more of software development. Such activities are frequently referred to as *good software engineering practices*. Many are described and defined in standards and methodologies such as the Institute of Electrical and Electronics Engineers (IEEE) Software Engineering Standards and the Software Engineering Institute's Capability Maturity Model Integration (CMMI) models. In this TIR, the activities are referred to as the tools that support the conclusion that the software is validated. A list and description of known tools can be found in Annex A.

3.3 Critical thinking

This TIR promotes the use of critical thinking to determine which activities should be performed to adequately validate specific software. Critical thinking is a process of analyzing and evaluating various aspects of software, as well as the environment in which it will be used, to identify the most meaningful set of confidence-building activities to be applied during validation. Critical thinking avoids an approach that applies a one-size-fits-all validation solution without thoroughly evaluating the solution to determine if it indeed results in the desired outcome. Critical thinking recognizes that validation solutions may vary greatly from software to software and also allows for different validation solutions to be applied to the same software in a similar situation. Critical thinking challenges proposed validation solutions, to ensure that they meet the intent of the regulation, and considers all key stakeholders and their needs. Critical thinking is also used to reevaluate the validation solution when characteristics of the software change, when the software's intended use changes, or when new information becomes available.

Critical thinking results in a validation solution that establishes compliance for a manufacturer, ensures that the software is safe for use, results in documented evidence that is deemed appropriate and adequate by reviewers, and results in a scenario in which individuals performing the validation work feel that the effort added value and represented a "least burdensome approach".

This TIR provides a framework in which key aspects of the software, such as intended use, risk, degree of reliance, and ability to determine if the software is working correctly, are evaluated to determine the most meaningful confidence-building activities that can be applied to the validation solution. This TIR describes a toolbox of activities from which to choose on the basis of the results of applying critical thinking.

4 Software validation and critical thinking

4.1 Overview

Throughout the life cycle of software for regulated processes, appropriate controls need to be in place to ensure that the software performs as intended. Incorporation of critical thinking and application of selected confidence-building activities result in establishing and maintaining a validated state of the software. Figure 1 depicts a conceptual view of typical activities and controls that are part of the life cycle from the moment the decision is made to automate a process until the software is retired or is no longer used for regulated processes. Although the figure depicts a sequential model, in reality the process is of an iterative nature as elements are defined, risks are identified, and critical thinking is applied.

Software for Regulated Processes

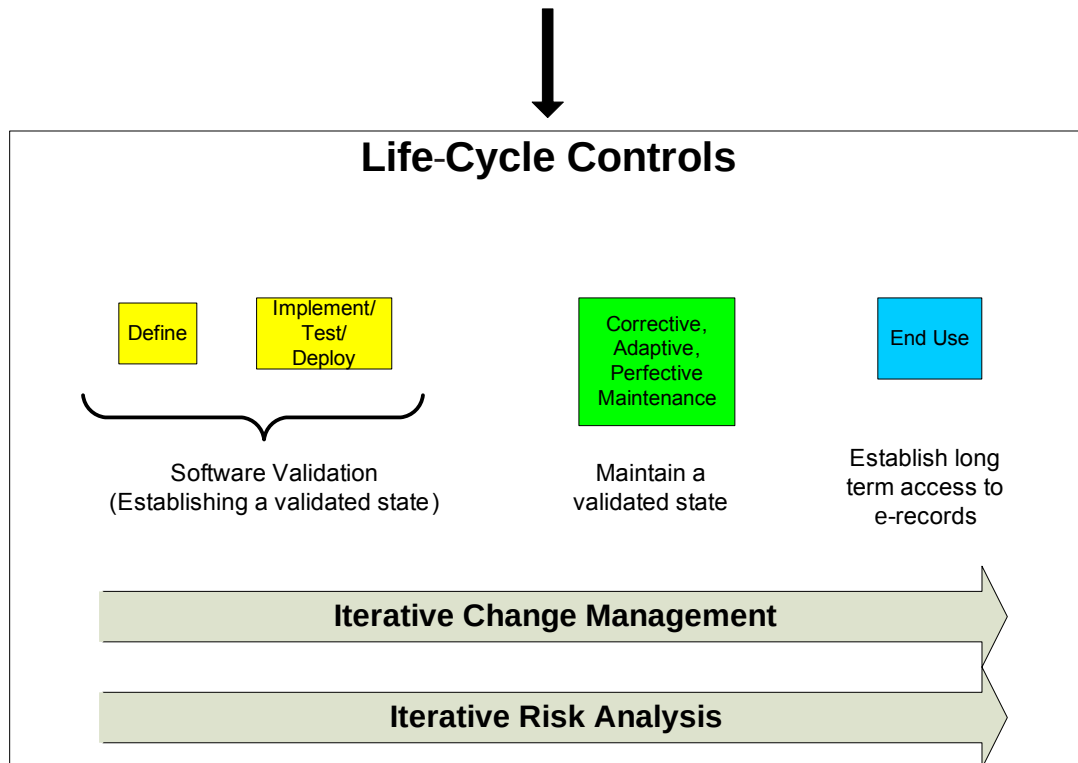


Figure 1—Life-cycle controls

A fundamental confidence-building activity to be selected from the toolbox is the choice of software development life-cycle model. The model chosen must include critical thinking processes and activities that enable the selection of other appropriate tools during various life-cycle activities. The results of the analyses and evaluations used drive the selection of the most meaningful set of confidence-building activities to ensure that the software performs as intended. This TIR does not mean to imply or prescribe the use of any particular software development model. For simplicity, however, the remainder of this document explains the concepts of critical thinking within the context of a waterfall development model, using generic names that the AAMI Validation of Software for Regulated Processes Task Group has chosen for the phases. Other software development models (e.g., iterative, spiral) can certainly be used as long as critical thinking and the application of appropriate tools are incorporated into the model.

When considering the automation of a process, one must identify whether the proposed software is automating part of a regulated process through an investigation of its intended use. If so, then the software must be validated for its intended use. Although this TIR describes an approach to validating software for regulated processes, the same approach is also good practice for software that is that is not used for regulated processes.

During the develop phase of the life cycle, risk management and validation planning tasks are performed to gather information and drive decisions in the following four areas:

- level of effort applied and scrutiny of documentation and deliverables,
- extent of content in the documentation and deliverables,
- selection of tools from the toolbox and methods for applying the tools, and
- level of effort in applying the tools.

The primary drivers for decisions in the four areas are process risk and software risk. However, other drivers may influence decisions, including the complexity of the software and process, the type of software, the software pedigree, and the control over the development of the software. Many of those elements can be affected by whether the software is internally developed, is vendor supplied, or is some combination of the two. The secondary drivers should be considered in the decisions to make or buy the software or its components.

The validation planning process consists of two distinct elements. The first validation planning element involves determining the level of rigor in the documentation and the scrutiny to be applied to the review of the resulting deliverables. The decisions in this element are primarily driven by the results of the process risk analysis. The second validation planning element drives the selection of tools from the toolbox to implement, test, and deploy the software. The choice of tools is driven primarily by the software risk analysis. Such planning steps result from different types of risk analyses and are depicted as separate activities in this report. However, many times the steps are combined into one activity, which includes the different aspects of risk analysis and the resultant choices for proceeding with validation.

During the *develop* phase of the life cycle, risk management and validation planning tasks are used to define the appropriate level of effort to be applied to the software and to determine what confidence-building tools to apply. This type of approach results in the completion of appropriate value-added activities and verification tasks, which are the basis for establishing a validated state. Once these activities and tasks are executed, the tools and their associated results are cited in a validation report as support for the conclusion that the software is validated.

Once deployed, the software moves into the *maintain* phase of the software life cycle. During this period, the software is monitored, enhanced, and updated as dictated by the business needs or regulatory requirement changes. Change management activities use the same concepts as the initial approach that was applied during the *develop* phase of the life cycle. Changes, however, are now assessed as to their effect on the intended use, on the risk of failure, on the risk control measures that were applied during the initial development, and on any functionality of the software itself. Once again, appropriate and value-added tools are selected during the definition, implementation, testing, and deployment of the changes. Applying such an approach ensures that the software continues to operate as intended in a validated state.

The *retire* phase is the act of removing software from use either by removal of the process being automated or by replacement of the software being used for the automated process. In either case, the software will no longer be operational for its intended use. Typically, the primary activities associated with retirement include the archiving of any data that are required to be maintained and the actual decommissioning of the software. It is essential to plan for methods of long-term data access depending on the record retention requirements defined by regulation.

The activities shown in Figure 1 reflect the primary software life-cycle control activities. Other work streams include project management, process development, vendor management (if applicable), and possibly others, depending on the software being implemented. The details of work streams are generally outside the scope of this TIR, but the work streams do interact and overlap with the life-cycle control activities. Therefore, work stream activities will be discussed when they directly affect or interact with the life-cycle control activities.

Figure 2 depicts software life-cycle control activities and critical thinking within the context of other work stream activities. The critical thinking activities appear in the iterative risk analysis and validation work streams. It is important to have clear and formal definitions of these work streams within the organization's business model to ensure that a program properly manages the software from both business and regulatory perspectives.

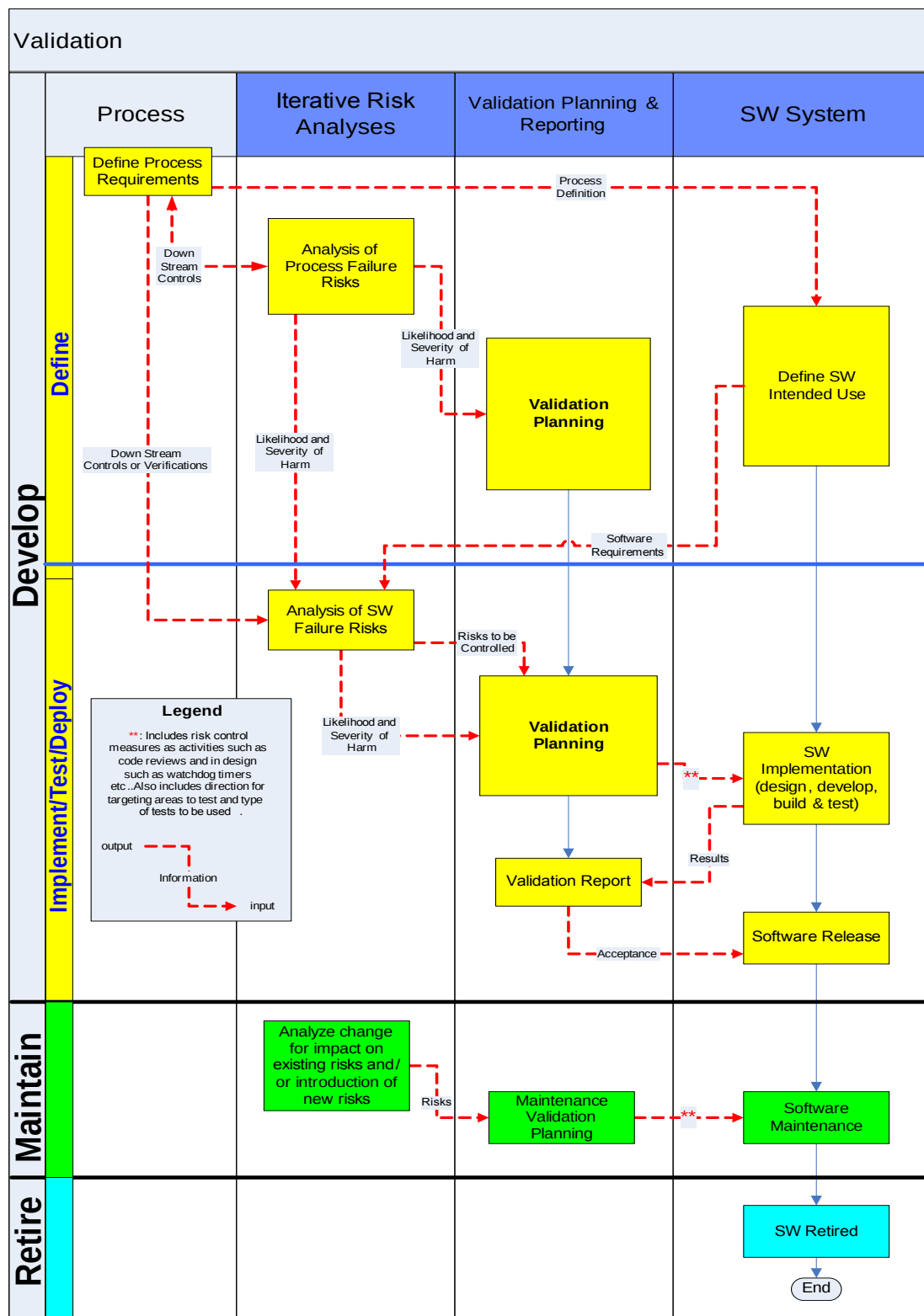


Figure 2—Life-cycle controls work streams

The various colors depicted in Figure 2 correspond to the life-cycle portion that is shown in the overall approach flow chart in Figure 1. The red dashed lines indicate information that is outputted from one activity and that provides input to or helps drive decisions in another activity. The diagram demonstrates how the ordering of the activities is driven

by the need to have input information before completing the activities that require the input. It is important to note that all the activities are completed irrespective of the size or complexity of the software being implemented. However, for larger or more complex software, such activities will most likely be discrete; for smaller or simpler software, many of those activities will be combined or completed simultaneously.

In summary, the critical thinking approach described in this TIR is a systematic method for identifying and including appropriate confidence-building activities or tools in various work streams to support the conclusions that the software is validated on release and that the validated state will be maintained until the software is retired.

The following sections provide additional detail for each of the blocks found in the life-cycle controls depicted in Figure 1. The sections use the work stream depiction of iterative risk analyses, validation, and software activities shown in Figure 2 to provide perspective on the various decision points and decision drivers that incorporate critical thinking.

4.2 In scope?

The first step in determining whether the software is considered to be used for regulated processes is to document a high-level definition of the process and use of the software. This activity may seem of small value when it is readily known that the software is in scope and one is already embarking on defining the full intended use of the software. However, for situations in which such assumptions are less clear, documenting the process and use enables the clear determination as to whether the software is in scope. In addition, for identified out-of-scope software, such an activity can result in a rationale as to why the software is out of scope.

Once the process to be automated has been identified and the expected, high-level intended use of the software has been defined, one can determine whether the software meets the definition of regulated process software by performing a regulatory use assessment. It is usually fairly straightforward to determine that software is in scope. Discerning out-of-scope software is more difficult. When a process contains elements both within and outside the scope of medical device regulations, the picture becomes blurred for software that automates both aspects. Hence, it is very important to clearly delineate the boundaries that define the use of software within a process. A related consideration, equally important, is to define those boundaries that differentiate regulated processes from nonregulated processes that are automated by modules of a highly complex software application.

Regulatory use assessment

A regulatory use assessment can be used to determine whether software meets the definition of “software for regulated processes” and therefore falls within the scope of this TIR. Start by identifying the specific regulations that apply to the processes that are automated by the software and the data records that are managed by the software. A series of questions can be used to help fully understand the role that the software plays in support of these regulations. The following types of questions should be considered:

- a) Could the failure or latent flaws of the software affect the safety or quality of medical devices?
- b) Does the software automate or execute an activity required by regulation (in particular, the requirements of the QSR)?
- c) Does the software generate or manage data to be used in or support of a regulatory submission?
- d) Does the software generate or manage records that are required by a regulation (e.g., device master record, device history record, design history file, or clinical trial records) or records that would be accessed in the future to provide evidence of the completion of an activity required by regulation?
- e) Is the software used to execute or record an electronic signature required by regulation?

The first two questions help to identify software used for regulated processes. The last three questions help to identify software that contains electronic records, electronic signatures, or both and that would also need to meet the appropriate Part 11 requirements. A “yes” answer to any of the questions identifies software that is required to be validated and is within scope of this TIR. The software identified through the last three questions should also meet the appropriate regulatory requirements for electronic records and signatures.

At times it can be difficult to determine whether a process and corresponding software are part of the quality system. Some tools can have many degrees of separation from the actual medical device. Each organization should, therefore, carefully consider the circumstances surrounding such borderline software and should completely understand the impact of the failure of the software on the regulated processes and, ultimately, on the safety and efficacy of any manufactured medical devices. When the answer is not certain, the best approach is to consider the software as in scope and to apply the approach defined in this TIR.

4.2.1 Processes and software extraneous to medical device regulations

When processes or software contain elements extraneous to medical device regulations, an analysis should be performed to determine which parts of the software are considered to be in scope and which parts are not in scope. Such decisions should be rationalized on the basis of the degree of integration between various components, modules, and data structures of the software and in accordance with the compliance needs of the organization. This rationalization is especially important in the case of software used in support of the quality system, such as large, complex enterprise resource planning (ERP) software. ERP software can include functionality for nonmedical device-regulated processes such as accounting and finance. Although such functionality may be crucial for business operations and may have to meet certain government requirements (e.g., those of the Sarbanes-Oxley Act), the functionality is extraneous to the medical device regulation and is not a concern to the FDA unless the functionality is also used to manage records required by regulation.

4.3 Develop phase

When critical thinking is applied during the *develop* phase, two primary validation planning activities capture the decisions made regarding validation effort and the selection of specific tools to be used. The first part of the validation planning activity uses input from the process risk analysis (see Annex B) to establish the basis for the level of effort that should be applied to the documentation and to drive the choices of tools from the *Define* section of the toolbox (see Annex A). At this point, the level of effort is defined as the extent of detail expected in the documentation and the extent of managerial/cross-functional involvement and independent reviews of the documentation. The second part of the validation planning activity uses input from the software risk analysis to drive the choices of the *implement*, *test*, and *deploy* tools from the toolbox. Level of effort at this stage means the risk-driven application of engineering risk control measures and choices through static and dynamic analyses. Once the activities are properly executed, the validated state of the software is established, and evidence of the validation is documented in the validation report.

Many development life-cycle models can be applied during the *develop* phase: iterative, spiral, and modified waterfall, to name a few. This TIR does not advocate or recommend any specific life-cycle model. However, it does expect a controlled methodology, based on the concept of defining requirements (e.g., intended use), to be used before implementation, testing, and deployment. This concept is fundamental to establishing the validation of software for its intended use.

4.3.1 Define

The activities completed within the *define* block include the definition of the process, the definition of the software intended use within that process, and the planning for the level of validation effort based on the inherent risks identified within the process being automated. Figure 3 depicts this portion of the *develop* phase within the selected waterfall model example.

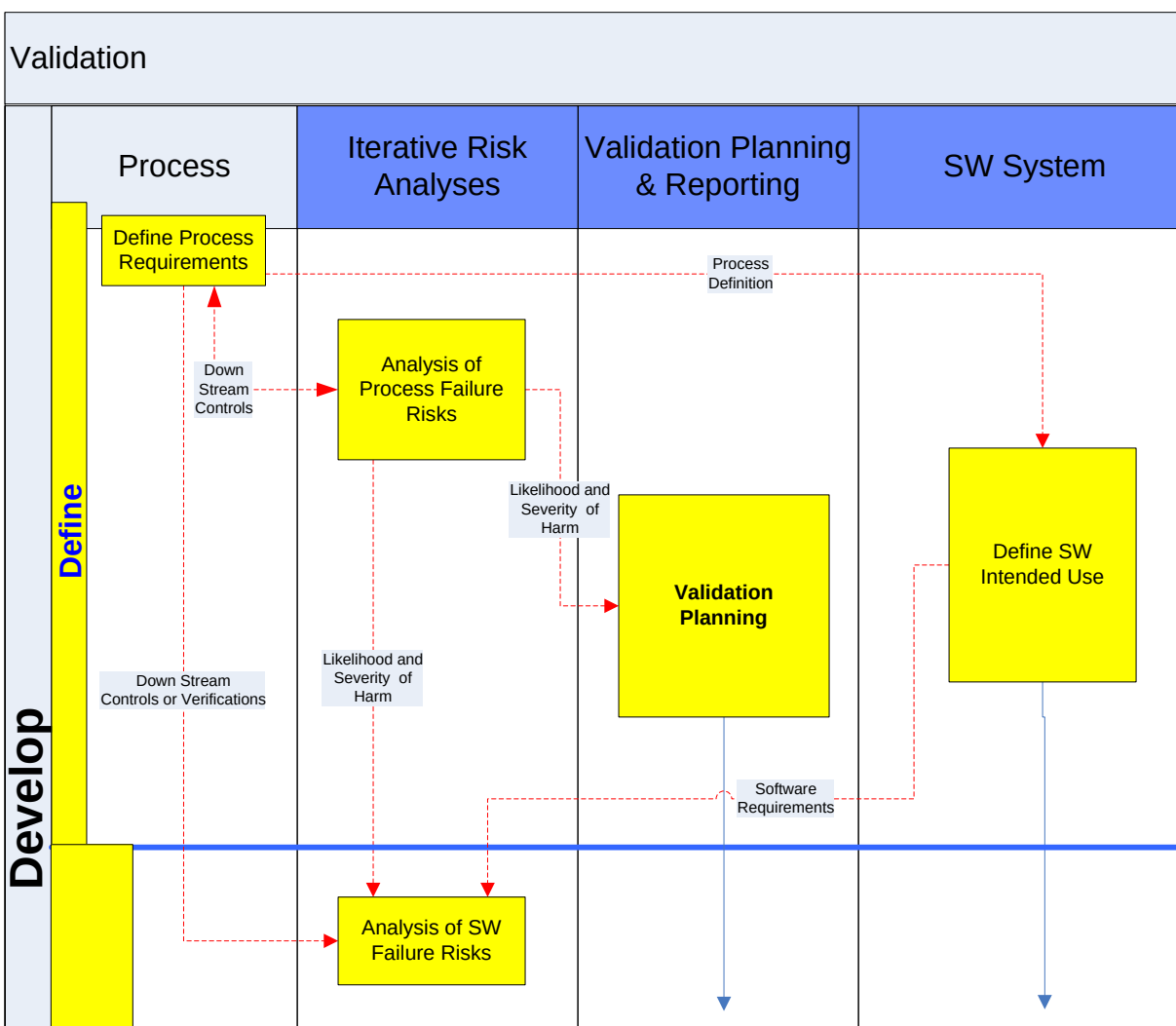


Figure 3—Life-cycle phase: Define block work streams

4.3.1.1 Process requirements

The first essential step in the application of life-cycle controls is to define clearly the purpose and function of the entire process, particularly the portions intended to be automated. Process definitions are best performed by involving the appropriate subject matter experts in the process to be automated. The complete definition includes all aspects and activities associated with the process regardless of whether all will be automated by the software. Benefits derived from this process definition include the following:

- Regulatory requirements can be clearly discerned.
- Intended use of the particular software within the context of the process can be clearly discerned.
- Process aspects and activities not automated by the particular software can be clearly identified and addressed procedurally or by some other means.
- Process activities upstream and downstream from the software are identified and can be considered when assessing the risks of the software failure and in devising risk controls for software failure.

The process definition activity establishes the foundation for decisions that are made later in the life cycle and is essential to targeting efforts on value-added, risk-based activities. The process definition activity should not be circumvented or minimized.

4.3.1.2 Analysis of process failure risk

When one is performing the risk analyses and discussing risk management in the context of software for regulated processes, the software has a relationship to the final safety and efficacy of the medical product (see discussion in Annex B). Several types of risks must be considered:

- **Risk of harm to humans.** This type refers to the risk of harm to users of the software, operators of equipment controlled by software, patients of the devices whose manufacture or quality is controlled by the software, and bystanders.
- **Regulatory risk.** This type covers the risk of noncompliance with regulatory requirements. It is important to consider if failure of the software can lead to loss of records (e.g., CAPA, complaint, device master record, or device history file records) required by regulatory agencies or to deviations from quality system and manufacturing procedures.
- **Environmental risk.** This type refers to the risk to the environment in which the software operates. Although typically thought of as risks related to toxic leak, spill, and fire hazards, environmental risk could also cover other types of fires, as well as floods, explosions, and the like. Additionally, the virtual environment should be considered. For example, could a failure of the software result in the loss or corruption of data used by other software?

Note that although FDA regulations do not address environmental or worker safety issues (which are regulated by separate government agencies), this TIR includes such factors in its description of risk management activities, given the potential impact from this type of software.

Other types of risks, such as project completion risks (e.g., funding for a project not in place) and business risk (e.g., business continuity) can be incorporated into this model. However, the scope of this TIR and the tools discussed to reduce risk do not address either project completion risks or business risks. This document focuses on the determination of the human safety risks, regulatory risks, and environmental risks associated with software failure within the context of process failure.

The analysis of process failure risk is intended to identify the harm that could occur as a result of process failure. Such an analysis should focus on the proposed process and should occur immediately after the completion of the process definition so that any identified risks can be considered when selecting a software solution, regardless of whether the software solution is internally developed or obtained externally. For example, when a process failure has a high risk of harm, one should choose a software solution that is based on technology that is well understood and predictable. When process failure does not pose a high risk of harm, there is less concern about spending significant time, energy, and money on more robust software solutions. The results of risk analysis should be clearly documented because such results are valuable decision drivers for selecting tools from the toolbox and for justifying the level of effort applied to the validation activities.

4.3.1.3 Validation planning

The extent of the *validation* (i.e., confirmation and objective evidence) needed to ensure that the requirements of the software can be consistently fulfilled depends on the critical value of the software within the overall process. Therefore, the first validation planning activity regarding the level of effort applied and the scrutiny of the deliverable elements is based solely on input from the process failure risk analysis.

Companies must resist falling into checklist mentalities instead of requiring good engineering judgment to define or specify the level of effort as driven by the potential risk of harm from the failure of the regulated process. Some software may only require a low level of effort such as defining the intended use, documenting the risk rationale, documenting the software functions per procedure (including confirmation that the software is consistent with these elements either through actions such as reviews or basic functional testing), and controlling the software configuration. A medium level of effort may require more detailed validation planning, multiple layers of intended use, and one or more verification test reports to establish the confidence that the appropriate level of validation has been performed. Software that has a high risk of causing harm may require a high level of rigor, including a full range of life-cycle control activities equivalent to those required for a medical device developed under design controls.

This validation planning activity results in a first iteration of validation planning documentation. The planning includes the selections for “level of effort” (i.e., the decisions) and the rationale for those choices (i.e., the decision drivers). The rationale should be based on the risk of harm posed by a failure of the regulated process. The validation plan should provide objective evidence of the application of critical thinking to the validation planning process.

4.3.1.4 Software intended use

The intended use of software comprises a linear progression of detail as driven by the risk and complexity of the software. The intended use is meant to provide a complete picture of the software functionality and its purpose within the process. Specifically, the intended use is meant to describe and explain how the software fits into the overall process that it is automating, what the software does, what one can expect of the software, and how much one can rely on the software to design, produce, and maintain safe medical devices. The intended use is a key tool used to understand what potential risks are associated with the use of the software.

The three main elements of intended use are as follows:

- **Purpose and intent.** Purpose and intent are related to
 - o the software's use (e.g., who, what, when, why, where, and how);
 - o the regulatory use of the software; and
 - o the boundaries of the software within the process or with other software and/or users.
- **Software use requirements.** As the complexity and, generally, the risk increases, this element adds more detailed information regarding the use of the software (e.g. use cases, user requirements).
- **Software requirements.** As the complexity and risk increase to the point where clear direction must be provided to implementers of the software, this element provides more specific and detailed information regarding the expectations of the software (e.g., Software Requirements Specification information as defined by IEEE).

The extent of documentation created for the intended use may vary depending on the complexity and size of the software. The intended use of simple software may be stated in a few sentences or paragraphs. In contrast, the intended use of more complex and higher-risk software may require the inclusion of extensive information across multiple documents.

The intended use should be formally controlled and approved. An organization should require the involvement of properly skilled and experienced personnel who have knowledge of the regulations, the quality system, and the process being automated. Larger-scale software or safety-critical software may also benefit from involving skilled and experienced personnel with knowledge of good software engineering practices and the expected software technology to be used.

Given that we must validate for “intended use,” validation cannot be accomplished unless the intended use for the software is sufficiently defined.

The following sections provide further detail about the elements of the software intended use.

4.3.1.4.1 Software purpose and intent

The software purpose and intent statement contains information covering three elements: software use, regulatory use, and boundary definitions. After completing the process of investigating the aspects of these elements (as described in this section), one should be able to generate a software purpose and intent statement that succinctly leads the reader to an understanding of the use of the software within the context of the Quality System.

Software use

When defining the use of the software, one should consider the following questions: what, why, how, who, where, and when. The answers to such questions help explore how the software is being used to meet the process requirements. Such exploration helps to identify base information for the definition of the software, as shown in Table 1.

Table 1—Sample questions and answers

Question	Answer
What problem is the software addressing?	There is a problem in efficiently and accurately pooling product defect data for trending purposes.
Why is the software useful?	The software enables the pooling and trending of data from global locations.
How does the software solve the problem?	The software drives the process of data collection and automatically pools and calculates trending information, or the software does not drive the process but provides a passive collection of data used to pool and calculate trending information.
Who uses the software?	The Quality Assurance and Operations Departments use the software.
Where is the software used?	The software is accessed by locations in the United States, Europe, and Japan.
When is the software used?	The software is accessed during normal business hours for the global locations (i.e., daily, Monday through Friday).

NOTE—The sample questions and answers provided here are not exhaustive.

As each of those questions is explored, the general importance of the answers becomes evident in determining the extent to which the software can influence a product's quality, the level of risk associated with the process, or both. The answers that are meaningful to the description of the software should be included in the established intended use definition.

Regulatory use

When evaluating the regulatory use, one can further explore the questions answered to determine if the software is in scope (see 4.2). Expand all “yes” answers to include the reasons for these conclusions. Now that the software has been identified as in scope, any potential harm to humans (other than the users of the medical device) or to the environment needs to be determined. All the following questions direct the user's consideration toward elements that are required as part of regulations (see section 2), such as public health, safety, and validity or authenticity of electronic records and signatures.

- How could the failure or latent flaws of the software affect the safety or quality of medical devices?
- How does the software automate or execute an activity required by regulation, in particular, the requirements of the Quality System Regulation?
- How does the software generate or manage data to be used in a regulatory submission or to support a regulatory submission?
- How does the software record or house any information for the device master record, device history record, design history file, or clinical trial records or information that would be accessed in the future to provide evidence of the completion of an activity required by regulation?
- How is the software used to execute or record an electronic signature required by regulation?
- How could the software cause harm to people (other than the users of the medical device) or to the environment?

Software boundaries

There are many benefits to identifying software boundaries. Defining the parts of the process that are to be automated through software (*boundaries within the process*) and the places where software interfaces exist (*boundaries with other software*) facilitates the effectiveness and efficiency of the validation efforts. For example, it often can be more efficient to validate multiple software products as a group rather than to perform individual validations. One should also consider how various grouping strategies can affect efficiencies of ongoing *maintain* phase activities.

Boundaries within the process

Identifying the boundaries of the software within the process to be automated clearly establishes the aspects that are to be included in the intended use. Software may automate an entire process or may automate a subset of activities within a process. Software may also function as a repository of data for the process (see Figure 4). Understanding the role that the software plays within the process helps to determine the risks associated with a potential failure of the software. If it is determined that the software meets the definition of software for regulated processes, understanding the software role and failure risks within the process will help to define the level of validation effort required. For example, software that automates an entire process and provides the only means to perform that

process may require a higher level of validation effort than software that automates only a minor piece of the process. Software that houses data crucial to proving the safety or efficacy of a device may require a higher level of effort than software containing data used for trending vendor performance.

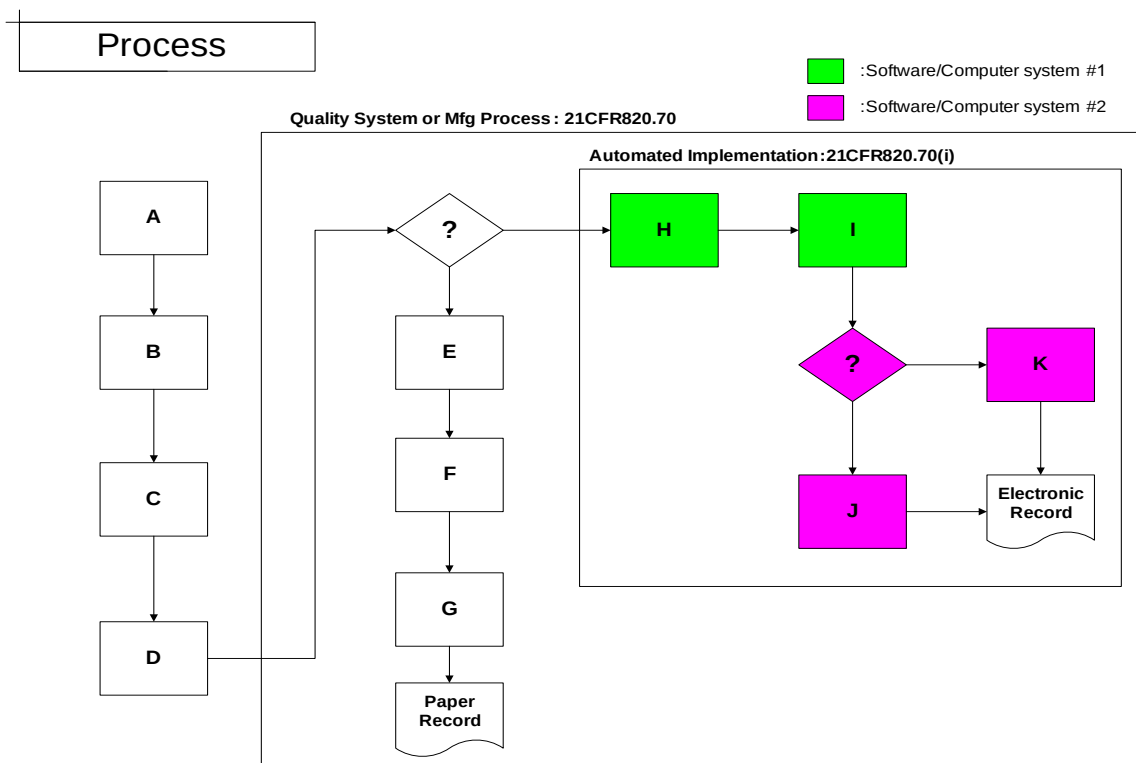


Figure 4—Depiction of software use and process boundaries

Boundaries with other software

Software may be externally interfaced with other regulated process software. When the boundaries of software with other software are defined, it is important to identify all interfaces between the applications. Validation efforts typically include the internal interfaces as an inherent part of the method. However, the software's external interfaces must not be ignored. Determining whether to include the external interfaces in the validation effort of the server application or the client application can be somewhat arbitrary and may depend on how project teams are set up to develop the various software pieces. In situations where a specific software application is used as the interface, it may make sense to treat the software as a stand-alone application with its own validation activities. In other circumstances, validation activities may be split between the two interfaced applications. In any case, all interfaces between software applications must be incorporated into the critical thinking process.

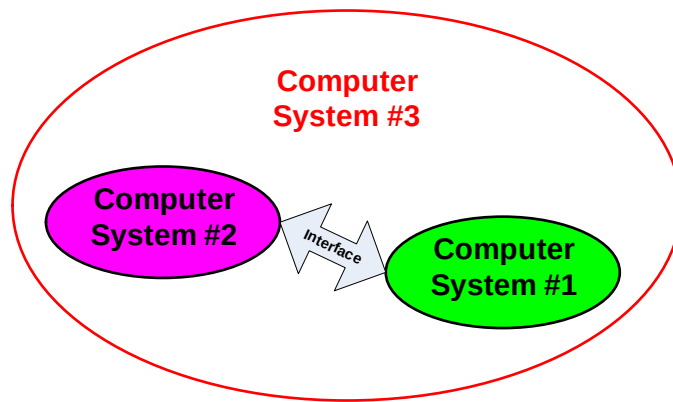


Figure 5—Depiction of software boundaries with other software

4.3.1.4.2 Software use requirements

Software use requirements consist of well-documented and traceable elements that provide an additional layer of detail about the use of the software compared with the software's purpose and intent. Such requirements provide insight into the use scenarios of the system, from either a user's perspective or a product-needs perspective. The user's perspective may be captured in the form of user requirements, use cases, or other user-centric definition of needs. The product-needs perspective captures the needs of the medical device that is being affected by the system and may, in some cases, include a reference to specific device requirements or a synopsis of the product lines that the software may affect. Those software use requirements provide more detailed input into the development of the software requirements.

4.3.1.4.3 Software requirements

Software requirements define specifically what the software needs to do in order to meet its purpose, intent and use requirements. These elements need to be well documented and traceable. The software requirements have varying levels of detail and involve different requirements definition approaches depending on the risk of the process and the origin of the system. Software requirements comprise the input to the system's design, to the system's configuration, as well as the input to testing activities.

4.3.2 Implement, test, and deploy

The activities completed within the *implement*, *test*, and *deploy* block include (a) planning of the level of validation rigor in the design, (b) development and configuration, (c) building of the software, and (d) testing of the software based on risks identified within the software itself. The decision to purchase software, rather than to develop it internally, will affect the types of tools selected from the toolbox. However, it is important to note that although the tools may be different, they still result in establishing confidence in the software. Again, the tools selected (decisions made) and the reasons for the tool selection (decision drivers) are documented during the validation planning activity. Once the controls are properly executed, and before the release of the software, the validated state is documented in the validation report. Figure 6 depicts this portion of the develop phase within the waterfall model example.

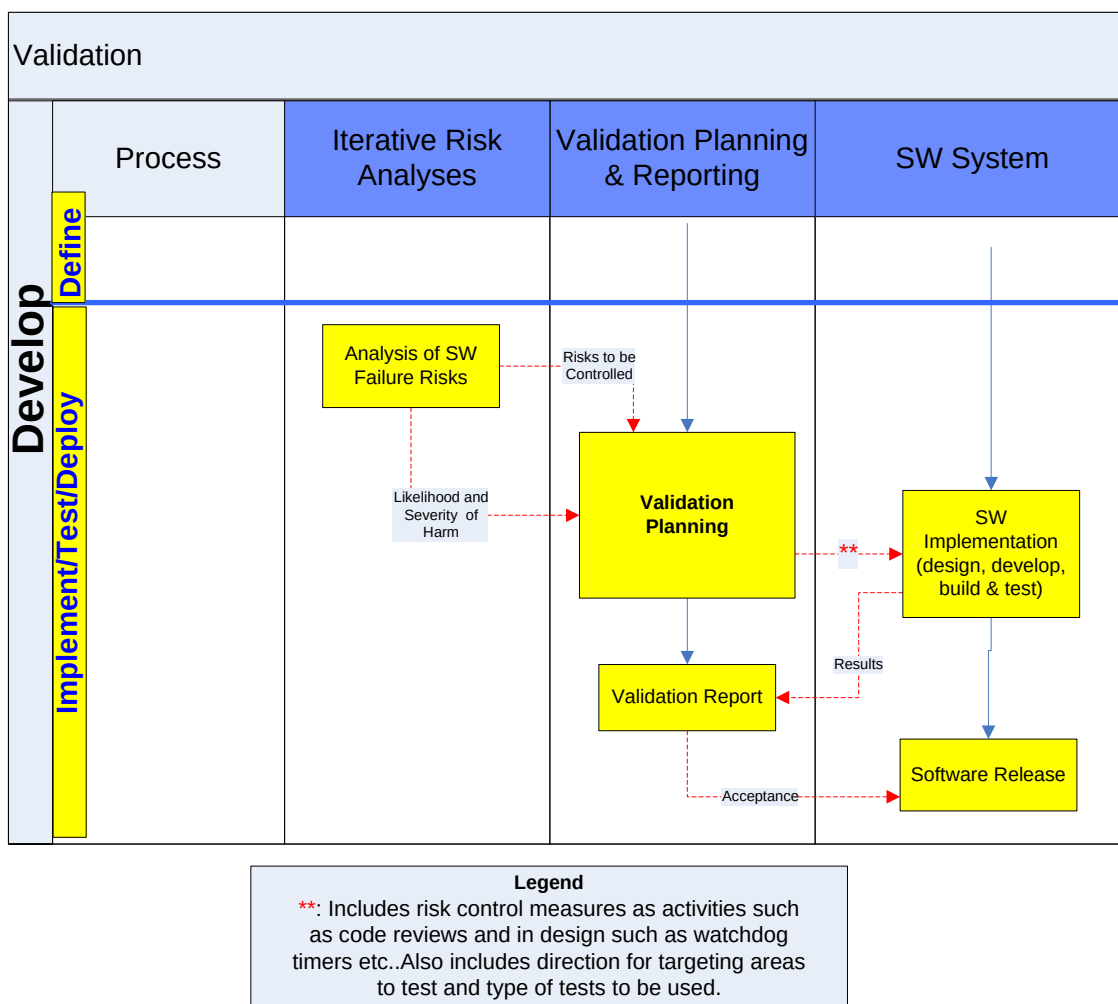


Figure 6—Life-cycle phase: *Implement, test, and deploy* block work streams

4.3.2.1 Analysis of software failure risks

As already discussed, there are two points during the overall process when risk analyses provide input to the validation planning steps. The first risk analysis is intended to analyze process failure; the second risk analysis is intended to identify the potential software failure risks and therefore can be completed only after a software solution has been chosen. The key point of software failure risk analysis is to determine and document the inherent risks associated with software failure and to identify any downstream control measures (including process and software controls). The analysis is then used to arrive at a realistic and effective validation approach.

When reviewing risks attributable to software failure, one must consider any downstream process controls that constitute risk control measures. Such risk control measures can reduce the impact of a software failure, thus reducing the dependency on the software and thereby reducing the reliance on testing (examination) and documentation (collection of objective evidence) to ensure the safe operation of the software. Also, one must consider that the act of moving a process from manual control to an automated process based on software inherently affects the risks associated with the process. Changes can introduce new risks, control existing risks, or affect the likelihood of risk occurrence. Including such considerations will help ensure that the software is viewed within the context of the overall process.

A model for performing software risk analysis is presented in Annex B. This model does not represent an all-encompassing formula. The resulting analysis provides input into the choice of tools from the toolbox to be used for the software validation. The quality of this analysis will be commensurate with the experience of the individuals

performing the analysis. This activity requires personnel with a significant amount of experience in software quality engineering.

4.3.2.2 Validation planning

The validation planning activity uses the intended use definition and the results of the software risk analysis as inputs to the identification of risk control measures and the selection of tools from the toolbox that will be used to validate the software.

The selection of specific tools will vary depending on the experience and skill sets of the individuals involved. It is important that the tool selection process include qualified individuals who have an understanding of the impact of failure on the process being automated. It is important that those individuals understand the process being automated and the inherent risks of failure of the software that will automate that process, although the individuals need not be software experts. Individuals from various disciplines (regulatory, quality, clinical, and the like) should be involved in the planning process for any software that is highly complex or that has a high risk associated with its failure.

The approach to validation may vary depending on the need to perform a phased release or releases for multiple instances of the applications that make up the system. For phased release expansion, use the maintenance methodologies discussed in 4.4 (e.g., a statistical process control system where validated functionality is expanded). For multiple instances, a base validation covering the common system elements may be performed and released. The base validation package would then be used as a foundation for each validation for a specific intended use (e.g., a base manufacturing execution system configured for different production lines).

It is not uncommon for an individual to be concerned about the result of a failure as it affects his or her own discipline yet to be totally unaware of the effects of the failure elsewhere. For example, manufacturing engineers may be concerned about risk to the production process (business risk) but may not have the insight to foresee how a manufacturing failure could manifest itself as a clinical safety risk. Based on the combination of skills and experiences of the validation planners, different planners may select different tools for the same software. As long as each solution meets the goal of building confidence in the performance of the software, each solution would be considered acceptable.

The validation planning activity results in a documented plan that describes the choices made (decisions) and the reasons for the choices being made (decision drivers). Validation planning provides documented evidence of the rationale used to select value-added confidence-building activities used to ensure that the software will perform as intended.

4.3.2.3 Software implementation (design, develop, build, and test)

The software implementation block includes the actual application of many tools from the toolbox and includes typical implementation activities that are defined by standard groups such as IEEE. As previously stated, a variety of development methodologies or life-cycle models can be used to effectively implement and deploy software. The activities described here depict a waterfall-type sequence only for the sake of simplicity. Iterative, spiral, and other valid life-cycle approaches may be utilized as long as the risk management, validation planning, and critical thinking concepts described in this TIR are applied within the context of the chosen life cycle.

4.3.2.4 Validation report

Once sufficient confidence-building activities, including tools selected from the toolbox, have been completed to ensure that the software performs as intended, the activities and (possibly) the results of the activities should be cited in a final validation report. The formal review and approval of the report provide a summary of references to all documented objective evidence that supports the conclusion that the software has been validated for its intended use.

4.3.2.5 Software release

Once a conclusion has been reached that the software performs as intended and does not introduce unacceptable risk into the regulated process, a formal controlled method must exist for releasing the software. The defined controls must ensure and confirm that the software placed into use matches the software that has been assessed through the confidence-building activities cited in the validation report. Alternatively, if the released software does not completely match the software that was validated (for reasons such as test simulation, hardware limitations, or other

environmental constraints), the rationales and controls must ensure and confirm that the results sufficiently represent the performance of the released software in its intended environment.

4.4 **Maintain phase**

After software is released for use in the final regulated process environment, it is in the *maintain* phase of its life cycle. *Maintain* phase activities consist of ensuring that the software remains in a validated state while accommodating, managing, and controlling various types of changes. Some types may involve only changes to the process within which the software is used.

Changes to any validated system should be done in a controlled manner in accordance with policies and procedures. Ideally, changes are made in a test environment and are validated before promoting the system to production use. When validating the change in a test environment is not possible and testing of changes must occur in the production environment, appropriate controls should be in place to minimize unwanted impacts to the production environment or directly to the product.

The selection of which tools from the toolbox are to be used in validating the change will be determined by the impact analyses of software changes on existing risk control measures, by the introduction of new risks, or by both. Additionally, since the actual use of the software or its configuration may migrate over time despite efforts to control it, the use of *maintain* phase-specific tools, such as periodic monitoring of actual use or real-time monitoring of software configuration, may be appropriate. If a change in intended use results in a higher risk level, the change can trigger a more extensive set of validation activities than was originally performed, even without changes to the software. Decisions regarding the choices and evidence of the execution of more extensive validation activities should be documented as part of validation planning to provide evidence that the software remains in a validated state.

4.4.1 **Planning for maintenance**

Ideally, maintenance planning begins during the *develop* phase. One must properly understand how changes will affect software validation, examine the effect of changes on risk, and plan the proper activities to maintain validation. Large and complex software may have to accommodate day-to-day maintenance and performance-tuning activities without affecting the software's ability to perform as intended. Planning for maintenance during the *develop* phase can define which of the operational activities can be done without affecting validation and which changes require validation efforts. Before the software arrives at the *maintain* phase, methods of determining when to perform further validation activities on the software should be planned and discussed, including how changes in an underlying software (e.g., operating system, database management system) might affect the validated software. It is helpful to train software operators to recognize such boundaries and to recognize the difference between normal operational activities and any changes requiring validation.

Traceability analysis is a useful tool in managing maintenance activities. Traceability analysis is frequently a cornerstone of the initial validation and is often facilitated through a traceability matrix. The matrix maps requirements for tests or other verification activities, risk control measures, and so forth. If performed well during the initial implementation, traceability analysis becomes a valuable tool during maintenance by facilitating the identification of the impact of changes and of the appropriate activities to validate the changes. In simple software, such analysis may consist of a single-level trace of requirements to implementation and verification. However, complex software may require a multilevel matrix that decomposes top-level functionality into lower-level requirements and then into implementation and verification. Other information may also be embedded—for example, sections of the software that are considered particularly high risk can be designated within the trace matrix, possibly with additional validation activities indicated.

4.4.2 **Types of maintenance within the *maintain* phase**

There are a number of reasons that software would change after it is released for use. Some of the more common types of maintenance changes include the following:

- corrective maintenance changes made to correct errors and faults in the software;
- perfective maintenance changes to enhance performance, maintainability or other software attributes; and
- adaptive maintenance done to update the software operational environment (e.g., changes to operating system or system hardware).

4.4.3 Process changes: Change to risk control measures

The process that is wholly or partially automated by software may change independently of the software. When a process change occurs, it is important to understand how that change affects the validated state of the software. The process change may affect the intended use of the software or of other supporting information regarding the software. The process change may also affect the risk control measures in place for the software that are part of the validation rationale. Because the software is part of a process, downstream controls may be important risk control measures for the software. If the downstream controls are properly identified as part of the software validation rationale and process definition, the impact analysis for the proposed process change will be easier to perform. Impact analysis is essential to performing maintenance in a way that builds confidence in both the software and the process within which the software is operating.

4.4.4 Emergency change

Software changes may need to be performed under emergency circumstances. Typically, such changes are required if the integrity of the software, operating system, or data has been compromised or to facilitate mitigation of potential harmful situations.

Emergency changes should be governed by approved processes. These processes should require the justification for development and implementation, the mechanisms for gaining and recording authorization to deploy the change, the provisions to ensure that the risk has been properly assessed and controlled, and any activities necessary to invoke the emergency change (e.g., training, communication, product review, and disposition). In this circumstance, performing the provisions for properly assessing and controlling risks represents the minimum set of activities needed to meet the regulatory requirement for the validation of changes prior to release. Additional, post-emergency change activities may be needed to fully evaluate all effects of the change. Depending on the overall risk posed by a failure of the process being automated, process output (data or product) may need additional controls until all post-emergency change activities are complete.

Software problems that interrupt an automated process are usually obvious. Detecting subtle, underlying problems can be more difficult. Periodic evaluation of error logs, help center requests, customer feedback, and other defect reports may point to underlying problems. Such monitoring techniques may pick up problems that are not obvious enough to result in an error report but that can indicate correctable software problems. Maintenance activities may then be necessary to deal with identified problems by implementing corrections in future releases. Additionally, issues in released software that are attributed to these types of software problems can be proactively managed.

After the maintenance activities correct the problems for future releases, the historical impact of identified defects in released software should be reviewed and their consequences managed.

If the software validation depends on ensuring the correct usage of the software through training, periodic evaluation of user training effectiveness is another monitoring technique that helps to maintain the validated state.

4.4.5 Maintaining for intended use

Change in intended use is a category that requires special attention because the change could be subtle and hard to detect or it could be quite obvious. In the subtle case, a change occurs to the purpose and intent or to the software use requirements and does not necessarily cause a change to the detailed software requirements element (see section 4.3.1.4). Such a change may occur intentionally or as a result of simply using existing software in a new mode without realizing that the intended use has been affected. Intended use may migrate over time, or users may start using the software in a way that was not originally intended. Because of this shift, the deployed software is no longer in a validated state. It must be validated for the new intended use, or the new use must cease. In the latter case, a risk assessment is in order to make sure no risks were introduced during the period of unauthorized use. Each time a change is initiated to validated software, the intended use should be reviewed to ensure that it is still consistent with the actual use of the software.

4.5 Retire phase

During the *retire* phase, the goal is to document the decommissioning of the software and to establish methods for accessing any associated electronic records throughout any required record retention periods.

Software retirement activities are highly dependent on the type of software being retired. Some software simply performs an activity and does not store any data. Other software can be as complex as a lot traceability or document control system, which houses volumes of product-related and compliance-related data. In the instance of software that stores data, a plan must exist for handling the data. Some issues to consider include the following:

- Is there software replacing the retired software?
- Can the data be migrated to the new software?
- Should the data be migrated to a portable format for long-term retention?
- What are the data retention requirements for the type of data?
- Will the data be stored on durable media?
 - If so, what are the storage instructions or procedures, and can the data be retrieved containing all associated data requirements?
 - What is the procedure for maintaining durable media and software that can read it?
 - Will an archived hardware platform be stored for using and retrieving the retired application?
 - How will stored hardware be maintained?
 - Would the retired software ever need to be accessed as part of a complaint or CAPA investigation?
 - Will the platform and application be needed to re-create a software program?

5 Documentation

It is essential to ensure that all information associated with the software life-cycle control activities is appropriately documented. Two major benefits arise from having high-quality and efficient documentation:

- 1) Complete software definition that is clearly articulated in documentation enables full understanding of the software's intended use and expected performance, and it enables the understanding of the full impact of any and all changes made to the software.
- 2) Record of validation planning and execution provides documented evidence of the decisions made as a result of critical thinking. Focusing this documentation around the evaluations or analyses performed and the resulting tool selections that target risk-based and meaningful confidence-building activities provides for a succinct understanding of the validation that was performed. With a summary of how the acceptance criteria were met, the documentation provides evidence that the activities completed ensure the software performs as intended and introduces acceptable levels of risk to the process it automates.

The extent of documentation produced is directly related to the level of effort applied to the validation of the software. The level of effort should be commensurate with risk. Therefore, the software validation approach discussed in this TIR bases the extent of documentation on the impact of the automated process failure. The greater the risk of harm to persons or to the environment posed by the automated process, the greater the extent of documentation expected. In addition, the higher risk of harm should drive a higher level of scrutiny of the documentation by multiple cross-functional peers, by higher levels of management within the company, or by both.

The organization of the life-cycle control information into documentation can vary depending on many factors, such as the technology used and the size or complexity of the software. The information should be organized in a manner that facilitates the auditing of the information along with the ability to maintain evidence of a validated state during the *maintain* phase of the software life cycle. How life-cycle control information is captured and documented depends on preferences and established policies of the parties performing the validation. Discretion is given to the parties validating the software regarding how the objective evidence of life-cycle controls is packaged and presented in documentation. From a compliance review perspective, the validation planning and reporting documentation should be established to provide a compilation of all value-added, confidence-building activities that were planned and executed in order to ensure that the software performs as intended. Essentially, this documentation is the key record of the choices made (decisions) on the basis of inputs (decision drivers) that embody the critical thinking process used to confirm that a complete software solution has been developed that meets the intent of the regulation and considers all key stakeholders and their needs.

NOTE—The term “documentation” is used to refer to the body of information that is recorded, whether it is recorded in an actual document or in tools that capture the information, such as requirements management tools.

6 Prerequisite processes

For the methodology discussed in this TIR to provide the most benefit, a solid quality system is crucial. The aspects of a quality system that can have the most positive effect on the success of the critical thinking methodology include asset and infrastructure management (human and hardware), change management (including configuration management), and vendor management. Detailing those aspects is outside the scope of this TIR; each aspect is addressed in other standards and documents within industry (see Annex E). In addition, this TIR does not intend to

associate specific roles or functions (e.g., quality assurance, management, and manufacturing) with the activities in this document. Each company's philosophy and human resource infrastructure will dictate the acceptable roles for performing validation activities.