

AD-A279 199

2



PROCESS DEFINITION AND MODELING GUIDEBOOK

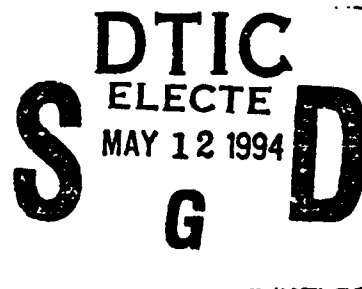
VOLUME 2

ADVANCED APPLICATIONS OF MPDM

SPC-92041-CMC

VERSION 02.00.02

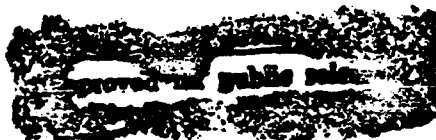
MARCH 1994



94-13725



42315



94 5 05 1 67

PROCESS DEFINITION AND MODELING GUIDEBOOK

VOLUME 2

ADVANCED APPLICATIONS OF MPDM

SPC-92041-CMC

VERSION 02.00.02

MARCH 1994

DTIC QUALITY INSPECTED 8

~~19950119~~ 054

PROCESS DEFINITION AND MODELING GUIDEBOOK

VOLUME 2

ADVANCED APPLICATIONS OF MPDM

SPC-92041-CMC

VERSION 02.00.02

MARCH 1994

Produced by the
SOFTWARE PRODUCTIVITY CONSORTIUM SERVICES CORPORATION
under contract to the
VIRGINIA CENTER OF EXCELLENCE
FOR SOFTWARE REUSE AND TECHNOLOGY TRANSFER

SPC Building
2214 Rock Hill Road
Herndon, Virginia 22070

Copyright © 1992, 1994, Software Productivity Consortium Services Corporation, Herndon, Virginia. Permission to use, copy, modify, and distribute this material for any purpose and without fee is hereby granted consistent with 48 CFR 227 and 252, and provided that the above copyright notice appears in all copies and that both this copyright notice and this permission notice appear in supporting documentation. This material is based in part upon work sponsored by the Defense Advanced Research Projects Agency under Grant #MDA972-92-J-1018. The content does not necessarily reflect the position or the policy of the U. S. Government, and no official endorsement should be inferred. The name Software Productivity Consortium shall not be used in advertising or publicity pertaining to this material or otherwise without the prior written permission of Software Productivity Consortium, Inc. SOFTWARE PRODUCTIVITY CONSORTIUM, INC. AND SOFTWARE PRODUCTIVITY CONSORTIUM SERVICES CORPORATION MAKE NO REPRESENTATIONS OR WARRANTIES ABOUT THE SUITABILITY OF THIS MATERIAL FOR ANY PURPOSE OR ABOUT ANY OTHER MATTER, AND THIS MATERIAL IS PROVIDED WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND.

Permission to Reprint

The process guidebook examples used in Section 6.3.5 consist of copyrighted material and are reprinted by permission of Vitro Corporation.

Interleaf is a trademark of Interleaf, Inc.

Microsoft Project is a registered trademark of Microsoft Corporation.

Paradox is a trademark of Borland International.

PROCESS WEAVER is a registered trademark of Cap Gemini Innovation.

Statemate is a registered trademark of i-Logix, Inc.

CONTENTS

PREFACE	xxvii
ACKNOWLEDGMENTS	xxix
1. INTRODUCTION	1-1
1.1 Purpose and Scope	1-1
1.1.1 A Practical Approach to Process Definition	1-1
1.1.2 Process Definition Goals	1-2
1.1.3 Goal-Driven Approach	1-3
1.2 Intended Audience and Uses	1-4
1.2.1 Scope of Use	1-4
1.2.2 Levels of Use	1-5
1.3 Guidebook Organization	1-5
1.4 Author Interaction and Feedback	1-7
1.5 Typographic Conventions	1-7
2. THE ROAD TO A DEFINED PROCESS	2-1
2.1 Change	2-1
2.1.1 Organizational Change	2-1
2.1.2 Organizing for Software Process Improvement	2-3
2.1.3 Roles in Organizational Change	2-3
2.1.4 Evolutionary Change Process	2-4
2.1.5 Starting With the Evolutionary Spiral Process	2-5

2.2 Introducing Process Definition Into Your Organization	2-7
2.2.1 Technology Transfer	2-8
2.2.2 Understand Your Purposes and Their Implications	2-9
2.2.3 Process Definition and Modeling Staff	2-11
2.2.4 Organizing for Process Definition Activities	2-11
2.2.4.1 Creating Process Definition	2-11
2.2.4.2 Verification and Validation	2-12
2.2.4.3 Process Definition Management	2-12
2.2.4.4 Process Definition "Process" Management	2-12
2.2.5 Getting Started	2-12
2.2.6 Ongoing Improvements	2-13
2.2.6.1 Process Improvement	2-13
2.2.6.2 Representation Improvement	2-13
2.2.7 Guidelines for Managing Process Definition	2-14
2.2.8 Process Engineering	2-16
2.2.9 External Entities and Relationships	2-17
2.3 The Scope of Process Definition	2-18
2.3.1 Forms of Process Definitions	2-18
2.3.2 Architecture, Activities, and Methods	2-18
2.3.3 Evolution of Process Definitions	2-20
2.3.4 Advanced Process Description Uses	2-21
2.3.4.1 Automated Enactment	2-21
2.3.4.2 Simulation	2-21
2.4 Using Metrics to Steer Toward Success	2-21
2.4.1 Important Characteristics	2-22
2.4.2 Incorporating Measurement in Process Definition	2-23
2.4.3 Metric Support for Process Tracking and Cost Modeling	2-23

2.5 Summary	2-24
3. ORGANIZATIONAL PROCESS DEFINITION	3-1
3.1. Goals of Organizational Process Definition	3-1
3.2. Top-Down Software Process Definition	3-2
3.3 Process Engineering	3-3
3.3.1 Process Engineering Process	3-3
3.3.2 Engineer Standard Process Assets	3-5
3.3.2.1 Organizational Process Definition	3-5
3.3.2.2 Define Business Area Process Definition(s)	3-6
3.3.2.3 Define Program Process Definition(s)	3-6
3.3.3 Engineer Project Process	3-7
3.3.3.1 Project Process Definition	3-7
3.3.3.2 Instantiate Process Plan	3-8
3.3.3.3 Enact Project Process	3-9
3.4 Process Architecture	3-9
3.4.1 Purposes	3-9
3.4.2 What Is Needed in an Architecture	3-10
3.4.3 Issues	3-11
3.5 The Pockets of Excellence Approach	3-12
3.5.1 Identify High-Value, Low-Risk Process Kernel	3-12
3.5.2 Identify Projects for Pocket of Excellence	3-12
3.5.3 Interactively Define Common Process Kernel for Target Projects	3-13
3.5.4 Pilot Instantiation of Process Kernel on Selected Project	3-13
3.5.5 Build Valid Process Definition	3-14
3.5.6 Transfer Validated Process Kernel to Other Projects in Pocket of Excellence	3-14
3.5.7 Select Next Process Kernel	3-14
3.5.8 Select Next Pocket of Excellence	3-15

3.5.9 Distill Process Kernel Definition(s) for Multiple Pockets of Excellence	3-15
3.5.10 Abstract Up to Organization-Wide Process Policies	3-15
4. ADVANCED TEMPLATES	4-1
4.1 Advanced Templates Overview	4-1
4.2 Advanced Templates	4-4
4.2.1 Foundation Template	4-4
4.2.1.1 Review History (Group of Fields)	4-4
4.2.1.2 Access Permissions (Group of Fields)/Tier 2	4-5
4.2.1.3 Project Invocation ID/Tier 4	4-5
4.2.2 Activity Templates Meta-Class	4-5
4.2.2.1 Objective/Tier 2	4-5
4.2.2.2 Abstraction Level	4-5
4.2.2.3 Activity Criteria and Process (Group of Fields)	4-6
4.2.2.4 Formal Activity Criteria and Process (Group of Fields)	4-6
4.2.2.5 Related Products (Group of Fields)	4-6
4.2.2.6 Related Supports (Group of Fields)	4-8
4.2.2.7 Risks (Group of Fields)	4-9
4.2.2.8 Constraints (Group of Fields)	4-10
4.2.2.9 Activity State Information (Group of Fields)	4-11
4.2.2.10 Process Quality Attributes (Group of Fields)/Tier 3	4-11
4.2.2.11 Duration Information (Group of Fields)/Tier 4	4-12
4.2.2.12 Planned and Actuals (Group of Fields)	4-12
4.2.3 Activity/Decision Templates/Tier 2	4-13
4.2.3.1 Source of Information, Planned (Group of Fields)	4-13
4.2.3.2 Source of Information, Actual (Group of Fields)	4-14
4.2.3.3 Minimum Threshold for Decision (Group of Fields)	4-14

4.2.4 Activity/Production Templates/Tier 2	4-14
4.2.4.1 Project Charge Code/Tier 4	4-14
4.2.5 Activity/Milestone Templates/Tier 2	4-15
4.2.6 Activity/Overhead Templates/Tier 3	4-15
4.2.7 Activity/Corrective Action Templates/Tier 3	4-15
4.2.8 Activity/Quality Assurance Templates/Tier 3	4-15
4.2.8.1 Type/Tier 3	4-15
4.2.9 Activity/Configuration Management Templates/Tier 3	4-15
4.2.9.1 Type/Tier 3	4-16
4.2.10 Activity/Audit Templates/Tier 3	4-16
4.2.11 Activity/Process Improvement Templates/Tier 3	4-16
4.2.11.1 Type/Tier 3	4-16
4.2.12 Activity/Training Templates/Tier 3	4-17
4.2.13 Activity/Purchasing Templates/Tier 3	4-17
4.2.14 Activity/Service Templates/Tier 3	4-17
4.2.15 Activity/Documentation Templates/Tier 3	4-17
4.2.16 Activity/Measurement Templates/Tier 3	4-17
4.2.16.1 Type/Tier 3	4-17
4.2.16.2 Collection/Tier 3	4-17
4.2.16.3 Recorded In/Tier 3	4-18
4.2.17 Product Templates Meta-Class	4-18
4.2.17.1 Quality Rating Levels (Group of Fields)	4-18
4.2.17.2 Product State Information (Group of Fields)	4-18
4.2.17.3 Related Activities (Multiple Occurrence)	4-19
4.2.17.4 Risks (Group of Fields)	4-19
4.2.17.5 Constraints (Group of Fields)	4-20
4.2.17.6 Source and Destination (Group of Fields)	4-20

4.2.18 Product/Tangible Product Template	4-20
4.2.18.1 Type/Tier 3	4-21
4.2.18.2 Security Classification/Tier 2	4-21
4.2.19 Product/Tangible Product/Contract Deliverable Template	4-21
4.2.20 Product/Tangible Product/Contract Nondeliverable Template	4-21
4.2.21 Product/Tangible Product/Noncontract Nondeliverable Template	4-21
4.2.22 Product/Tangible Product/Status Reports	4-21
4.2.23 Product/Intangible Product Template/Tier 2	4-22
4.2.24 Product/Intangible Product/Knowledge Templates/Tier 3	4-22
4.2.25 Product/Intangible Product/Skill Templates/Tier 3	4-22
4.2.25.1 Proficiency Rating Levels (Group of Fields)	4-22
4.2.26 Support Templates Meta-Class	4-23
4.2.26.1 Type of Access Available/Tier 3	4-23
4.2.26.2 Share (Group of Fields)	4-23
4.2.26.3 Security Clearance/Tier 2	4-23
4.2.26.4 Supported Activities (Multiple Occurrence)	4-24
4.2.26.5 Efficiency Rating (Group of Fields)/Tier 4	4-24
4.2.26.6 Current Level of Support Efficiency/Tier 4	4-24
4.2.26.7 Risks (Group of Fields)	4-25
4.2.26.8 Support State Information (Group of Fields)	4-25
4.2.27 Support/Role Templates	4-26
4.2.27.1 Reports To/Tier 2	4-26
4.2.27.2 Reported To By/Tier 2	4-26
4.2.27.3 Unique Individual/Tier 4	4-26
4.2.27.4 Proficiency (Group of Fields)/Tier 3	4-26
4.2.27.5 Internal Constraint Waiver Authority/Tier 3 (Group of Fields)	4-26
4.2.27.6 Approved Staff Unique ID (Multiple Occurrence)/Tier 4	4-27

4.2.28 Support/Tool Templates/Tier 2	4-27
4.2.29 Support/Resource Templates	4-27
4.2.29.1 Owned By/Tier 3	4-27
4.2.30 Constraint Templates Meta-Class/Tier 2	4-27
4.2.30.1 Constrained Activities (Multiple Occurrence)/Tier 2	4-28
4.2.30.2 Constrained Products (Multiple Occurrence)/Tier 2	4-28
4.2.30.3 Constrained Supports (Multiple Occurrence)/Tier 2	4-28
4.2.30.4 Constraint Induced Risks (Group of Fields)/Tier 2	4-28
4.2.31 Constraint/Internal Constraint Template	4-28
4.2.31.1 Waiver Criteria/Tier 2	4-28
4.2.31.2 Roles Authorized to Exercise Waiver (Multiple Occurrence)/Tier 2 ..	4-28
4.2.32 Constraint/External Constraint Template	4-28
4.2.33 Risk Template Meta-Class/Tier 2	4-28
4.2.33.1 Severity Levels (Group of Fields)/Tier 2	4-29
4.2.33.2 Activity Risks Unique IDs (Multiple Occurrence)/Tier 2	4-29
4.2.33.3 Product Risks Unique IDs (Multiple Occurrence)/Tier 2	4-29
4.2.33.4 Support Risks Unique IDs (Multiple Occurrence)/Tier 2	4-29
4.2.33.5 Constraint Risks Unique IDs (Multiple Occurrence)/Tier 3	4-29
4.2.34 Risk/Cost Risk Template	4-29
4.2.35 Risk/Quality Risk Template	4-29
4.2.35.1 Type of Quality Risk/Tier 3	4-29
4.2.36 Risk/Schedule Risk Template	4-30
4.2.37 Risk/Performance Risk Template	4-30
4.2.37.1 Performance Risk Applies To/Tier 3	4-30
4.2.38 Risk/Construction Risk Template	4-30
4.2.39 Risk/Predictability Risk Template	4-30
4.3 Summary	4-31

5. ALTERNATIVE PROCESS REPRESENTATIONS	5-1
5.1 Characteristics of Process Representation Notations	5-1
5.1.1 Scalability	5-1
5.1.2 Applicability	5-1
5.1.3 Flexibility	5-2
5.1.4 Readability	5-2
5.1.5 Maintainability	5-2
5.1.6 Learnability	5-2
5.1.7 Robustness	5-2
5.1.8 Relative Formality	5-3
5.1.9 Representative Power	5-3
5.1.10 Characteristics of "Best" Representations	5-3
5.2. Degrees of Formality	5-4
5.3. Choosing the Right Process Representation Notation	5-5
5.3.1 Environments	5-5
5.3.2 Resources	5-6
5.3.3 Budget Constraints	5-6
5.3.4 History	5-6
5.3.5 Goals	5-6
5.3.6 Organizational Issues and Process Characteristics	5-7
5.4. Alternative Models and Representations	5-8
5.4.1 State Transition Diagrams	5-9
5.4.2 Entry-Task-Validation-eXit	5-10
5.4.3 Structured Analysis and Design Technique	5-12
5.4.4 Statecharts	5-14
5.4.5 Petri Nets	5-18
5.4.6 Process and Artifact State Transition Abstraction	5-20

5.4.6.1 Structure of Formal Generic Process Model	5-20
5.4.6.2 Artifacts and Their States	5-21
5.4.6.3 Process State	5-21
5.4.6.4 Translating From Process Templates to Process and Artifact State Transition Abstraction	5-22
5.4.7 Role Interaction Nets	5-23
5.5. Benefits to the Template-Based Process Representation	5-25
5.5.1 Scalability	5-26
5.5.2 Applicability	5-26
5.5.3 Flexibility	5-26
5.5.4 Readability	5-26
5.5.5 Maintainability	5-27
5.5.6 Learnability	5-27
5.5.7 Robustness	5-27
5.5.8 Formality	5-27
5.5.9 Final Remarks	5-28
APPENDIX A. INSPECTION TEMPLATES EXAMPLE	A-1
A.1 Peer Review Activity Templates	A-1
A.1.1 Activity Template: Peer Reviews	A-2
A.1.1.2 Unique ID	A-2
A.1.1.3 Brief Description	A-2
A.1.1.4 Overview Description	A-2
A.1.1.5 Summary Description	A-3
A.1.1.6 Parent Template(s)	A-3
A.1.1.7 Child Templates...	A-4
A.1.1.8 Used Within...	A-6
A.1.1.9 Transition Text...	A-6

A.1.1.10 Activity Criteria and Process...	A-6
A.1.1.11 Related Products...	A-7
A.1.1.12 Related Supports...	A-8
A.1.2 Activity Template: Inspection Planning	A-9
A.1.2.1 Name	A-9
A.1.2.2 Unique ID	A-9
A.1.2.3 Brief Description	A-9
A.1.2.4 Overview Description	A-9
A.1.2.5 Summary Description	A-9
A.1.2.6 Parent Template(s)	A-9
A.1.2.7 Child Templates...	A-9
A.1.2.8 Used Within...	A-9
A.1.2.9 Transition Text...	A-10
A.1.2.10 Activity Criteria and Process...	A-10
A.1.2.11 Related Products...	A-11
A.1.2.12 Related Supports...	A-12
A.1.3 Activity Template: Inspection Overview	A-14
A.1.3.1 Name	A-14
A.1.3.2 Unique ID	A-14
A.1.3.3 Brief Description	A-14
A.1.3.4 Overview Description	A-14
A.1.3.5 Summary Description	A-14
A.1.3.6 Parent Template(s)	A-14
A.1.3.7 Child Templates...	A-14
A.1.3.8 Used Within...	A-14
A.1.3.9 Transition Text...	A-15
A.1.3.10 Activity Criteria and Process...	A-15

A.1.3.11 Related Products...	A-15
A.1.3.12 Related Supports...	A-16
A.1.4 Activity Template: Inspection Preparation	A-17
A.1.4.1 Name	A-17
A.1.4.2 Unique ID	A-17
A.1.4.3 Brief Description	A-17
A.1.4.4 Overview Description	A-17
A.1.4.5 Summary Description	A-17
A.1.4.6 Parent Template(s)	A-17
A.1.4.7 Child Templates...	A-17
A.1.4.8 Used Within...	A-17
A.1.4.9 Transition Text...	A-18
A.1.4.10 Activity Criteria and Process...	A-18
A.1.4.11 Related Products...	A-18
A.1.4.12 Related Supports...	A-19
A.1.5 Activity Template: Inspection Meeting	A-20
A.1.5.1 Name	A-20
A.1.5.2 Unique ID	A-20
A.1.5.3 Brief Description	A-20
A.1.5.4 Overview Description	A-20
A.1.5.5 Summary Description	A-20
A.1.5.6 Parent Template(s)	A-20
A.1.5.7 Child Templates...	A-20
A.1.5.8 Used Within...	A-21
A.1.5.9 Transition Text...	A-21
A.1.5.10 Activity Criteria and Process...	A-21

A.1.5.11 Related Products...	A-22
A.1.5.12 Related Supports...	A-23
A.1.6 Activity Template: Inspection Meeting Review	A-24
A.1.6.1 Name	A-24
A.1.6.2 Unique ID	A-24
A.1.6.3 Brief Description	A-24
A.1.6.4 Overview Description	A-24
A.1.6.5 Summary Description	A-25
A.1.6.6 Parent Template(s)	A-25
A.1.6.7 Child Templates...	A-25
A.1.6.8 Used Within...	A-25
A.1.6.9 Transition Text...	A-25
A.1.6.10 Activity Criteria and Process...	A-25
A.1.6.11 Related Products...	A-26
A.1.6.12 Related Supports...	A-26
A.1.7 Activity Template: Inspection Meeting Reinspection Decision	A-27
A.1.7.1 Name	A-27
A.1.7.2 Unique ID	A-27
A.1.7.3 Brief Description	A-27
A.1.7.4 Overview Description	A-27
A.1.7.5 Summary Description	A-27
A.1.7.6 Parent Template(s)	A-27
A.1.7.7 Child Templates...	A-27
A.1.7.8 Used Within...	A-27
A.1.7.9 Transition Text...	A-27
A.1.7.10 Activity Criteria and Process...	A-27

A.1.7.11 Related Products...	A-28
A.1.7.12 Related Supports...	A-28
A.1.8 Activity Template: Inspection Rework	A-29
A.1.8.1 Name	A-29
A.1.8.2 Unique ID	A-29
A.1.8.3 Brief Description	A-29
A.1.8.4 Overview Description	A-29
A.1.8.5 Summary Description	A-29
A.1.8.6 Parent Template(s)	A-29
A.1.8.7 Child Templates...	A-29
A.1.8.8 Used Within...	A-29
A.1.8.9 Transition Text...	A-29
A.1.8.10 Activity Criteria and Process...	A-30
A.1.8.11 Related Products...	A-30
A.1.8.12 Related Supports...	A-31
A.1.9 Activity Template: Inspection Follow-Up	A-32
A.1.9.1 Name	A-32
A.1.9.2 Unique ID	A-32
A.1.9.3 Brief Description	A-32
A.1.9.4 Overview Description	A-32
A.1.9.5 Summary Description	A-32
A.1.9.6 Parent Template(s)	A-32
A.1.9.7 Child Templates...	A-32
A.1.9.8 Used Within...	A-32
A.1.9.9 Transition Text...	A-32
A.1.9.10 Activity Criteria and Process...	A-33

A.1.9.11 Related Products...	A-33
A.1.9.12 Related Supports...	A-34
A.1.10 Activity Template: Causal Analysis	A-36
A.1.10.1 Name	A-36
A.1.10.2 Unique ID	A-36
A.1.10.3 Brief Description	A-36
A.1.10.4 Overview Description	A-36
A.1.10.5 Summary Description	A-36
A.1.10.6 Parent Template(s)	A-36
A.1.10.7 Child Templates...	A-36
A.1.10.8 Used Within...	A-36
A.1.10.9 Transition Text...	A-37
A.1.10.10 Activity Criteria and Process...	A-37
A.1.10.11 Related Products...	A-37
A.1.10.12 Related Supports...	A-38
A.2 Peer Review Product Templates	A-40
A.2.1 Product Template: Inspection Input Product	A-41
A.2.1.1 Name	A-41
A.2.1.2 Unique ID	A-41
A.2.1.3 Brief Description	A-41
A.2.1.4 Overview Description	A-41
A.2.1.5 Summary Description	A-41
A.2.1.6 Parent Template(s)	A-41
A.2.1.7 Child Templates...	A-41
A.2.1.8 Used Within...	A-41
A.2.1.9 Transition Text...	A-41
A.2.1.10 Related Activities	A-41

A.2.2 Product Template: Review Memos	A-42
A.2.2.1 Name	A-42
A.2.2.2 Unique ID	A-42
A.2.2.3 Brief Description	A-42
A.2.2.4 Overview Description	A-42
A.2.2.5 Summary Description	A-42
A.2.2.6 Parent Template(s)	A-42
A.2.2.7 Child Templates... ..	A-42
A.2.2.8 Used Within... ..	A-43
A.2.2.9 Transition Text... ..	A-43
A.2.2.10 Related Activities	A-43
A.2.3 Product Template: Inspection Invitation Memorandum	A-44
A.2.3.1 Name	A-44
A.2.3.2 Unique ID	A-44
A.2.3.3 Brief Description	A-44
A.2.3.4 Overview Description	A-44
A.2.3.5 Summary Description	A-44
A.2.3.6 Parent Template(s)	A-44
A.2.3.7 Child Templates... ..	A-44
A.2.3.8 Used Within... ..	A-44
A.2.3.9 Transition Text... ..	A-45
A.2.3.10 Related Activities	A-45
A.2.4 Product Template: Inspection Results Memorandum	A-46
A.2.4.1 Name	A-46
A.2.4.2 Unique ID	A-46
A.2.4.3 Brief Description	A-46
A.2.4.4 Overview Description	A-46

A.2.4.5 Summary Description	A-46
A.2.4.6 Parent Template(s)	A-46
A.2.4.7 Child Templates...	A-46
A.2.4.8 Used Within...	A-46
A.2.4.9 Transition Text...	A-47
A.2.4.10 Related Activities	A-47
A.2.5 Product Template: Review Metrics	A-48
A.2.5.1 Name	A-48
A.2.5.2 Unique ID	A-48
A.2.5.3 Brief Description	A-48
A.2.5.4 Overview Description	A-48
A.2.5.5 Summary Description	A-48
A.2.5.6 Parent Template(s)	A-48
A.2.5.7 Child Templates...	A-49
A.2.5.8 Used Within...	A-49
A.2.5.9 Transition Text...	A-49
A.2.5.10 Related Activities	A-49
A.3 Peer Review Role Templates	A-50
A.3.1 Role Template: Peer Review Team	A-51
A.3.1.1 Name	A-51
A.3.1.2 Unique ID	A-51
A.3.1.3 Brief Description	A-51
A.3.1.4 Overview Description	A-51
A.3.1.5 Summary Description	A-51
A.3.1.6 Parent Template(s)	A-51
A.3.1.7 Child Templates...	A-51
A.3.1.8 Used Within...	A-53

A.3.1.9 Transition Text...	A-53
A.3.1.10 Supported Activities	A-53
A.3.1.11 Reports To...	A-53
A.3.1.12 Reported To By...	A-53
A.3.2 Role Template: Moderator	A-54
A.3.2.1 Name	A-54
A.3.2.2 Unique ID	A-54
A.3.2.3 Brief Description	A-54
A.3.2.4 Overview Description	A-54
A.3.2.5 Summary Description	A-54
A.3.2.6 Parent Template(s)	A-54
A.3.2.7 Child Templates...	A-54
A.3.2.8 Used Within...	A-54
A.3.2.9 Transition Text...	A-54
A.3.2.10 Supported Activities	A-55
A.3.2.11 Reports To...	A-55
A.3.2.12 Reported To By...	A-55
A.3.3 Role Template: Scribe	A-56
A.3.3.1 Name	A-56
A.3.3.2 Unique ID	A-56
A.3.3.3 Brief Description	A-56
A.3.3.4 Overview Description	A-56
A.3.3.5 Summary Description	A-56
A.3.3.6 Parent Template(s)	A-56
A.3.3.7 Child Templates...	A-56
A.3.3.8 Used Within...	A-56
A.3.3.9 Transition Text...	A-56

A.3.3.10 Supported Activities	A-57
A.3.3.11 Reports To...	A-57
A.3.3.12 Reported To By...	A-57
A.3.4 Role Template: Reader	A-58
A.3.4.1 Name	A-58
A.3.4.2 Unique ID	A-58
A.3.4.3 Brief Description	A-58
A.3.4.4 Overview Description	A-58
A.3.4.5 Summary Description	A-58
A.3.4.6 Parent Template(s)	A-58
A.3.4.7 Child Templates...	A-58
A.3.4.8 Used Within...	A-58
A.3.4.9 Transition Text...	A-58
A.3.4.10 Supported Activities	A-59
A.3.4.11 Reports To...	A-59
A.3.4.12 Reported To By...	A-59
A.3.5 Role Template: Inspectors	A-60
A.3.5.1 Name	A-60
A.3.5.2 Unique ID	A-60
A.3.5.3 Brief Description	A-60
A.3.5.4 Overview Description	A-60
A.3.5.5 Summary Description	A-60
A.3.5.6 Parent Template(s)	A-60
A.3.5.7 Child Templates...	A-60
A.3.5.8 Used Within...	A-61
A.3.5.9 Transition Text...	A-61
A.3.5.10 Supported Activities	A-62

A.3.5.11 Reports To...	A-62
A.3.5.12 Reported To By...	A-62
A.3.6 Role Template: Developer	A-63
A.3.6.1 Name	A-63
A.3.6.2 Unique ID	A-63
A.3.6.3 Brief Description	A-63
A.3.6.4 Overview Description	A-63
A.3.6.5 Summary Description	A-63
A.3.6.6 Parent Template(s)	A-63
A.3.6.7 Child Templates...	A-63
A.3.6.8 Used Within...	A-63
A.3.6.9 Transition Text...	A-63
A.3.6.10 Supported Activities	A-64
A.3.6.11 Reports To...	A-64
A.3.6.12 Reported To By...	A-64
A.3.7 Role Template: Key Inspector	A-65
A.3.7.1 Name	A-65
A.3.7.2 Unique ID	A-65
A.3.7.3 Brief Description	A-65
A.3.7.4 Overview Description	A-65
A.3.7.5 Summary Description	A-65
A.3.7.6 Parent Template(s)	A-65
A.3.7.7 Child Templates...	A-65
A.3.7.8 Used Within...	A-65
A.3.7.9 Transition Text...	A-65
A.3.7.10 Supported Activities	A-66

A.3.7.11 Reports To...	A-66
A.3.7.12 Reported To By	A-66
A.3.8 Role Template: Regular Inspector	A-67
A.3.8.1 Name	A-67
A.3.8.2 Unique ID	A-67
A.3.8.3 Brief Description	A-67
A.3.8.4 Overview Description	A-67
A.3.8.5 Summary Description	A-67
A.3.8.6 Parent Template(s)	A-67
A.3.8.7 Child Templates... ..	A-67
A.3.8.8 Used Within... ..	A-67
A.3.8.9 Transition Text... ..	A-67
A.3.8.10 Supported Activities	A-68
A.3.8.11 Reports To... ..	A-68
A.3.8.12 Reported To By	A-68
A.3.9 Role Template: Review Coordinator	A-69
A.3.9.1 Name	A-69
A.3.9.2 Unique ID	A-69
A.3.9.3 Brief Description	A-69
A.3.9.4 Overview Description	A-69
A.3.9.5 Summary Description	A-69
A.3.9.6 Parent Template(s)	A-69
A.3.9.7 Child Templates... ..	A-69
A.3.9.8 Used Within... ..	A-69
A.3.9.9 Transition Text... ..	A-70
A.3.9.10 Supported Activities	A-70

A.3.9.11 Reports To...	A-70
A.3.9.12 Reported To By	A-70
APPENDIX B. CHECKLISTS	B-1
B.1 Activity Template	B-3
B.2 General Process Definition Checklist	B-5
B.2.1 Attractiveness for Adoption	B-6
B.2.2 Role Usage	B-7
B.2.3 Definition Properties	B-10
B.2.4 Adaptability	B-13
B.2.5 Detailed Contents	B-15
B.2.6 Competition	B-23
LIST OF ABBREVIATIONS AND ACRONYMS	Abb-1
GLOSSARY	Glo-1
REFERENCES	Ref-1
BIBLIOGRAPHY	Bib-1

FIGURES

Figure P-1. Structure for Integrated Application of Consortium Technologies	xxvii
Figure 2-1. Organizational Subsystems and Context	2-2
Figure 2-2. Process Improvement Process	2-4
Figure 2-3. The Conceptual Evolutionary Spiral Process Model: A Management Focus ..	2-6
Figure 2-4. An Elaborated Evolutionary Spiral Process Model: A Complete Process Focus	2-7
Figure 2-5. Success Factors in Process Transfer	2-8
Figure 2-6. Uses of Process Definitions	2-10
Figure 2-7. Eleven-Step Scale for How Organizations React to Risks or Need to Improve	2-15
Figure 2-8. Typical Elements of an Activity	2-19
Figure 2-9. Process Definition Evolution	2-20
Figure 3-1. Abstraction Levels for Process Definitions	3-4
Figure 4-1. Template Inheritance Model	4-2
Figure 5-1. Entry-Task-Validation-eXit Diagram	5-10
Figure 5-2. Structured Analysis and Design Technique Diagram	5-12
Figure 5-3. Structured Analysis and Design Technique Example 1	5-13
Figure 5-4. Structured Analysis and Design Technique Example 2	5-14
Figure 5-5. Statechart Example 1	5-15
Figure 5-6. Statechart Example 2	5-16
Figure 5-7. Petri-Type Net	5-19
Figure 5-8. Relationships Defining the Design Model	5-21
Figure 5-9. P-State Diagram Example	5-22

Figure 5-10. Artifact Relation Diagram Example	5-23
Figure 5-11. Role Interaction Net	5-24
Figure 5-12. Role Interaction Net Templates	5-25

TABLES

Table P-1. Consortium Guidebooks and Related Practices	xxviii
Table 1-1. Guidebook Audience	1-4

PREFACE

The technology described in this guidebook is part of a broad approach to software productivity improvement. This preface provides an overview of that approach and identifies the series of guidebooks that support it. These guidebooks were developed by the Software Productivity Consortium under contract to the Virginia Center of Excellence for Software Reuse and Technology Transfer (VCOE). For a complete listing of VCOE guidebooks and products, call the Software Productivity Consortium's Technology Transfer Clearinghouse at (703) 742-7211.

Each technology has been packaged so it can be used without reference to the other technologies. However, it is also possible to combine these technologies into an integrated approach for product development. Figure P-1 shows how the guidebooks for these technologies relate to the practices of software development organizations.

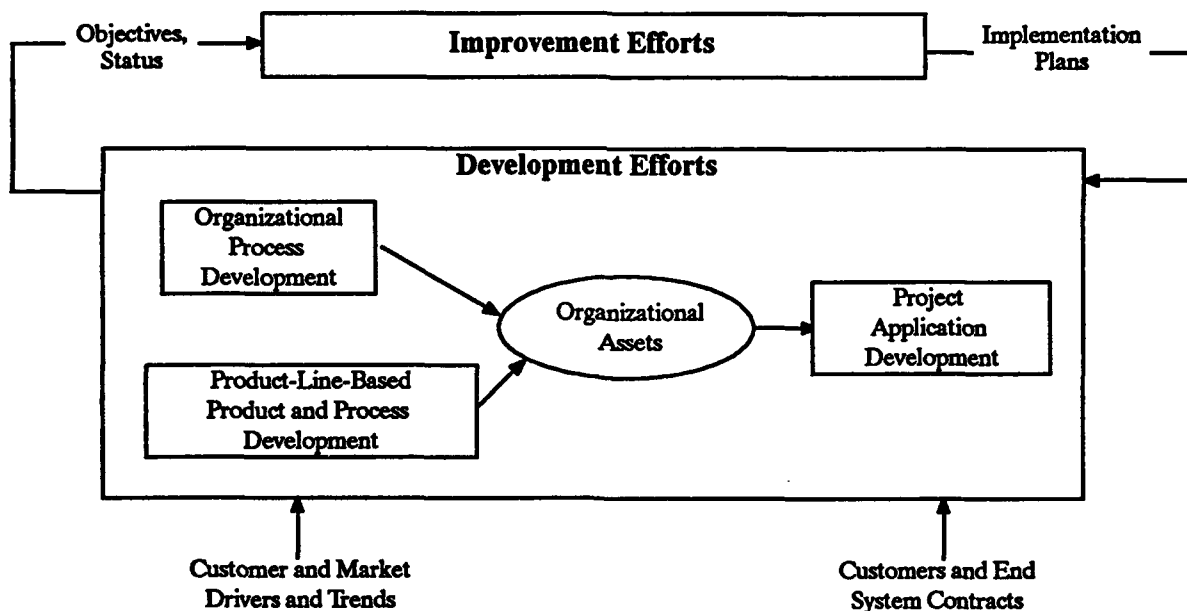


Figure P-1. Structure for Integrated Application of Consortium Technologies

These practices are composed of:

- **Improvement Efforts (IE).** Application of technology to improve software development efforts. These efforts require managed approaches to assessment of objectives and current capabilities, planning for the improvement, implementation of the plan, and measurement of success.
- **Development Efforts.** Development of products that meet the needs of customers and markets or products that make the organization more competitive in meeting expected future needs.

- **Organizational Process Development (OPD).** Development of standardized organizational process assets (e.g., process and method descriptions, process enactment tools) tailored for a particular organization.
- **Product-Line-Based Product and Process Development (PLD).** Development of integrated product and process assets (e.g., core products and processes for adapting them for particular customer needs) appropriate for a particular product line.
- **Project Application Development (PAD).** The tailoring and application of organizational assets for a particular product development effort.

Table P-1 describes how existing products can be integrated to address your organizational practice.

Table P-1. Consortium Guidebooks and Related Practices

Guidebook	Part Number	Relationship to Software Practice
<i>Consortium Requirements Engineering Guidebook</i>	SPC-92060-CMC	Used for defining and analyzing requirements in PAD. Adaptable for use in PLD.
<i>Managing Process Improvement: A Guidebook for Implementing Change</i>	SPC-93105-CMC	Supports IE by providing a process and supporting guidance for initiating and maintaining an organizational process improvement program.
<i>Process Definition and Modeling Guidebook</i>	SPC-92041-CMC	Provides methods for defining and documenting processes so they can be analyzed, modified, and enacted. Supports IE and OPD.
<i>Process Engineering With the Evolutionary Spiral Process Model</i>	SPC-93098-CMC	Used to iteratively plan, manage, and control PAD and PLD. Used to construct organization-specific processes in PLD and tailor them in PAD.
<i>Reuse Adoption Guidebook</i>	SPC-92051-CMC	Supports IE by providing specific process improvement activities for incorporating reuse practices.
<i>Reuse-Driven Software Processes Guidebook</i>	SPC-92019-CMC	Provides development approaches for PLD (domain engineering) and PAD (application engineering) of reusable software assets.
<i>Software Measurement Guidebook</i>	SPC-91060-CMC	Supports IE by providing methods for quantitative assessment of project status.
<i>Using New Technologies: A Technology Transfer Guidebook</i>	SPC-92046-CMC	Supports IE by providing a process that addresses how to get an organization to use new technologies.

This guidebook considerably expands upon the first version to reflect lessons and examples from use, and to increase coverage in a number of areas. For example, the number of templates has more than doubled, the number of data fields has increased nearly tenfold, and the definition of process risks and process metrics has been given explicit support. Additionally, template usage has been separated into four tiers of formality, thereby allowing you to easily select and use only those templates and fields that directly support your process definition goals.

ACKNOWLEDGMENTS

Volume 1 of this guidebook was authored by Richard Bechtold and John Brackett. Volume 2 was authored by Richard Bechtold, John Brackett, and Sam Redwine.

The Consortium would like to thank the Vitro Software Engineering Process Group members: Walt Greenspon, Rick Emery, Pauline Martin, Herb Nachmann, and Ed Zarrella and the Vitro Corporation for their participation in a process definition pilot project. Valuable insights were gained—and numerous improvements made—in the Consortium's process definition methodology and techniques as a direct result of their work.

The Consortium would also like to thank Alice Forey and Jennifer McLaughlin, SYSCON SEPG members. Their early use of the revised process templates likewise resulted in significant improvements to the material presented in this guidebook.

Thanks go to Consortium personnel Mary Eward, Donna Garfield, and Chuck Lynch for their extensive reviews and recommendations for improving this guidebook.

Special thanks go to Bobbie Troy, Consortium Technical Editor, whose efforts substantially improved the organization and presentation of this information. Thanks also go to Debbie Morgan and Deborah Tipeni for incorporating the markups to the draft document and to Betty Leach and Tina Medina for clean proofing the final copy.

This page intentionally left blank.

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

The objective of this guidebook is to guide you in efficiently developing and evolving a quality set of process definitions that will direct and improve the ways you develop software—both to improve your products and process and to allow assessment at the DoD Software Engineering Institute's (SEI) Capability Maturity Model (CMM) Levels 2 (Repeatable) and 3 (Defined). You should be better able to engineer your process descriptions and the processes themselves.

1.1.1 A PRACTICAL APPROACH TO PROCESS DEFINITION

This guidebook offers a practical approach to reflecting experience and distilled research, and includes highlights from a significant, real-world example. This approach has several characteristics that help ensure a successful process definition and modeling effort. For example, Managed Process Definition Methodology (MPDM):

- Provides “how to” guidance for early success and a sound foundation for continuing success
- Organizes the definition as it is developed
- Avoids unnecessarily elaborate or formal notations
- Is driven by your goals
- Considers your context and the factors you need to deal with in your situation
- Exploits simple automation to rapidly generate and revise guidebooks, training material, etc.
- Directly supports management (including self-management) of the process definition effort

Organizations need to have a representation option that combines the strengths of text- and graphic-based representations while minimizing their respective weaknesses. MPDM is specifically designed to support process representation as three separate yet tightly integrated concerns: (1) design, development, and implementation of the process model; (2) generation of various output products, such as guidebooks and training material, from your process model; and (3) management and review of the development of both the process model and the derived output products.

The data-organizing templates presented in this guidebook can be rendered as paper templates, but their design directly facilitates “automated templates” via mainstream information management repositories. Examples of template fields include text-oriented descriptions of activities, pre- and

post-conditions, internal processing, comments, role descriptions, product descriptions, risk types and levels, revision history, etc. However, through a variety of relationships the templates also convey an explicit architecture directly supporting graphical rendering and analysis. This combined template-and graphically-based approach provides you greater opportunity for the optimal combination of both text and diagrams toward the cost-effective development and use of process representations.

1.1.2 PROCESS DEFINITION GOALS

There are numerous goals for process definition:

- To improve your processes organization-wide such as through a Total Quality Management (TQM) initiative
- To reduce software management or technical problems
- To respond to customer or market pressures to improve or certify such as SEI CMM or ISO 9000
- To reach aggressive business goals involving software
- To augment or replace existing process documentation that is either unused or too expensive to maintain

Process definitions are needed for the same reasons sports teams need playbooks. A team without a playbook must transfer everything from head to head typically through long apprenticeship, is only as good as what is in their heads, has more trouble adjusting and improving plays, scratches their changes only in the dirt, and can never be on the same page. Though they do different things with them, playbooks are important to both coaches and players—an essential organizational asset.

Every software organization, regardless of size or maturity, has a process for developing and maintaining software. When using a defined software process, your organization may experience some of the following benefits:

- *Improved productivity (and teamwork)* because communication among the process users, managers, process developers, and customers is more effective
- *Reduced rework* because you identify and eliminate problems early in the process rather than later
- *Efficient project staff start-up time* because there is a documented process to train them on
- *Reduced software development costs* due to reduced volatility in software development processes
- *Improved predictability of budgets and schedules* because you have defined what to measure, when to measure it, and how to use the information
- *Improved tool usage* because tool usage is defined and supported by training material
- *Faster project start-up* because the project has a process that it can tailor

- *Increased integration between resulting products* due to improved coordination among and communication between teams
- *Increased quality of the resulting products* because reviews are defined and understood to be an integral part of the process

Existing examples of process representations include policy, procedures, and operational manuals developed by organizations to inform and guide their employees in the performance of their responsibilities. Most organizational process guidebooks only define the process, but a few make use of relatively high-level or simple process models.

Although process modeling is a comparatively rare technique for representing organizational processes, it is a well-known and mature technique for representing systems implemented by computer systems. Example techniques include Statecharts, Structured Analysis and Design Technique (SADTs), and the Entry-Task-Validation-eXit (ETVX) paradigm. Due to fundamental parallels between defining and modeling organizational processes and computer-based systems processes, you can apply many techniques from systems process representation to organizational process representation. Similarly, many of the advantages and benefits derived by building computer process representations can also be derived from organizational process representations.

1.1.3 GOAL-DRIVEN APPROACH

Unfortunately, probably as many attempts at process definition fail as succeed. The job is not simple or intuitive and can fail in a number of ways. These ways to fail include: undertaking the wrong scope; trying to define too much too soon; ignoring existing staff, process, or culture; over or under designing of process; using inappropriate techniques for process definition; developing an inconsistent or rigid definition; producing guidebooks that are not used; updating too slowly; and loss of sponsorship or support due to cost or schedule overruns within the process definition effort.

To avoid these types of failures, you need to clearly define specific goals. Example goals include:

- Analyze process (in)efficiencies
- Identify and remove process redundancies
- Identify and eliminate areas where the process is unknown or undefined
- Gain insights into process risk
- Identify where, when, and how process metrics will be collected and used

The key point is that it is important to allow your goals to drive your process definition effort. If the goal is to analyze the representation for process bottlenecks, this will influence the type of representation that should be constructed. If the goal is to identify and remove process redundancy, to eliminate areas where the process remains largely undefined, seek insights into process risk, or to establish or analyze a metrics program, all these considerations influence not only the type of definition or model constructed but also the approach taken in researching and constructing those representations. This issue is examined in detail in Section 5, Volume 2.

Process representation can span the effort spectrum from easy to exceedingly difficult as a function of process complexity and desired level of detail:

- Initially, you may find that relatively simple diagrams of isolated parts of your process are sufficient to achieve your immediate goals.
- Later, you may find that you can achieve additional goals by extending your existing representation, adding more detail, and maybe capturing primary interrelationships between your growing inventory of "process asset" models.
- Still later, other goals may be achieved through even further expansion of scope, detail, abstraction, and information.

The magnitude of your goals must be directly related to the magnitude of organizational support for process representation. This guidebook, and the tools and techniques it proposes, have been designed to facilitate precisely this type of goal-driven, incremental approach to process definition and modeling.

1.2 INTENDED AUDIENCE AND USES

The intended audience for this guidebook includes practitioners interested in developing process definition(s) that are acceptable for SEI Level 2 or 3 assessments, ISO-9000 certification, and those interested in the tangible benefit derived from applying process definition and modeling techniques and methods including automation. This audience is primarily:

- Software Engineering Process Group (SEPG) members
- Process Action Teams
- Process engineers and modelers

Line engineers, project managers, and anyone working on or interested in the area of process analysis, design, development, improvement, or management can also use this guidebook (see Table 1-1 a mapping of topics to audience). Because most of the intended readers are not necessarily familiar with the issues and aspects of process definition, representation, and modeling, motivation and rationale behind the recommended techniques are provided throughout the guidebook.

To clarify what sections you should read, Table 1-1 identifies generic audience types and the sections of this guidebook that each should read. Keep in mind that at any one time you may fit the description of more than one audience type.

Table 1-1. Guidebook Audience

Audience Type	Volume 1			Volume 2						
	1	2	3 - 6	1	2	3	4	5	A	B
Person doing basic process definition	✓	✓	✓	✓					✓	✓
Person leading basic definition	✓	✓	✓	✓	✓					
Person doing advanced definition	✓	✓	✓	✓		✓	✓	✓		
Person leading advanced definition	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Person performing process modeling			✓				✓			✓
Person controlling resources	✓	✓		✓	✓	✓				

1.2.1 SCOPE OF USE

This guidebook can help you:

- Establish a common foundation for process engineering, training, and documentation
- Develop process-oriented guidebooks, improve their usability, or reduce their cost
- Develop process training and education
- Improve your process either in general, for ISO 9000 certification, or to CMM Level 2 or 3
- Construct process representations either generally or based upon SADT, ETVX, or another process representation paradigm
- Model your processes

While this guidebook can be useful in a variety of contexts, it particularly benefits people who must develop a software process improvement action plan calling for definition of parts or all of their software processes.

This guidebook introduces both fundamental and advanced process definition concepts, explains how to develop a definition, and provides nontechnical as well as technical advice. While you can use this guidebook by itself, you are also provided with relevant references to other material.

1.2.2 LEVELS OF USE

This guidebook provides a flexible set of templates and techniques for capturing and representing processes and explains how to use the template-based information to produce guidebooks, training materials, and models. The material is presented on two primary levels:

- Basic usage: suitable for you if you need to produce process guidebooks and training material
- Advanced usage: suitable for those having increased need of process formality (to support, for instance, process simulation and automated enactment)

Volume 1 presents basic usage principles and techniques; Volume 2 presents advanced usage concepts. Volume 1 conveys the entire MPDM. Volume 2 provides more details and, in some areas, extends the scope. Although Volume 2 addresses issues related to automated process enactment, the principle subject of this guidebook is how to prepare process definitions that can be effectively used by people.

1.3 GUIDEBOOK ORGANIZATION

This guidebook is separated into two volumes. Volume 1 covers the basics suitable for organizations needing to produce process definitions intended to be enactable by humans as opposed to machines. Volume 1 contains Sections 1 through 6.

The structure of Volume 1 is:

- Section 1, Introduction, describes the goals, scope, and audience of this guidebook.
- Section 2, Process Definition Basics, presents the role of process definition within an organization and fundamental process definition concepts; it shows where and how you use process definitions and introduces MPDM.
- Section 3, MPDM Templates and Graphical Notation, describes the conceptual model behind the information gathering templates, goal-based process definition, and template “tiers of usage.” Then it discusses in detail the Tier 1 templates and their associated fields.
- Section 4, Fundamentals of Defining Your Process, explains how to start using the templates and describes in detail the eight tasks of MPDM.
- Section 5, Examples: Peer Review Process, presents a complete set of examples for MPDM-based process definition and modeling. These examples include a context diagram, indented lists, graphical models, completed templates, a navigation and extract algorithm, and an example of automatically generated pages to a guidebook.
- Section 6, Automated Process Definition and Modeling, discusses the use of mainstream, commercial off-the-shelf tools that support MPDM and briefly discusses a process definition pilot project.
- The List of Abbreviations and Acronyms contains abbreviations and acronyms used in this guidebook and their definitions.
- The Glossary contains a list of terms used in this guidebook and their definitions.
- The References section contains sources cited in this guidebook.
- The Bibliography section contains additional sources of information.

Volume 2 covers a more advanced approach suitable for those with prior process definition experience, focuses on process definition issues from the larger perspective of general process improvement, and contains information on defining processes that are enactable by machines. Volume 2 contains Sections 1 through 5 and two appendices.

The structure of Volume 2 is:

- Section 1, Introduction, describes the goals, scope, and audience of this guidebook.
- Section 2, The Road to a Defined Process: The Management Perspective, discusses organizational change, the introduction of process definition and modeling into an organization, the scope of organizational process definition, and briefly discusses metrics.
- Section 3, Organizational Process Definition, presents the goals of organizational process definition; a top-down approach to process definition; process engineering; process architectures; and an alternative, bottom-up, “pockets of excellence” approach to process definition.
- Section 4, Advanced Templates, presents the full set of process definition templates and describes Tier 2, Tier 3, and Tier 4 usage.

- Section 5, Alternative Process Representations, describes characteristics of process representations, degrees of formality, choosing the “right” process notation, alternative notations and models, and the benefits of MPDM.
- Appendix A, Inspection Templates Example, provides a completed set of templates to support the example in Section 5 of Volume 1.
- Appendix B, Checklists, provides a set of checklists that you can tailor to facilitate your process definition effort.
- The List of Abbreviations and Acronyms contains abbreviations and acronyms used in this guidebook and their definitions.
- The Glossary contains a list of terms used in this guidebook and their definitions.
- The References section contains sources cited in this guidebook.
- The Bibliography section contains additional sources of information.

1.4 AUTHOR INTERACTION AND FEEDBACK

The Consortium is actively interested in end-user reaction, case studies, feedback, pilot projects, and any suggestions or recommendations you have for improving and advancing the content of this guidebook. While insights from researchers and technologists are also appreciated, the Consortium is especially interested in feedback from those producing process definitions. The Consortium encourages you to apply the information presented, and strongly encourages feedback on how to make this information progressively more useful to you and others. Contact Software Productivity Consortium, SPC Building, 2214 Rock Hill Road, Herndon, Virginia 22070 (703) 742-7211. Ask for (or write to) the Process Definition Technology Manager in the Process Improvement Division.

1.5 TYPOGRAPHIC CONVENTIONS

This guidebook uses the following typographic conventions:

- Serif font General presentation of information.
- Initial Capitalization, Serif font Names of fields, values within fields, activities, and work products.
- CAPITALIZED SERIF font Names of undesired events.
- Italicized serif font* Mathematical expressions and publication titles.
- Boldfaced serif font** Section headings and emphasis.
- Boldfaced italicized serif font*** Run-in headings in bulleted lists and, in the Appendix, minor subsections.
- Sans serif font Information ordering.

This page intentionally left blank.

2. THE ROAD TO A DEFINED PROCESS

Having a successfully defined process consists of more than just having a definition. To remain useful, the definition must be accepted, trained and performed, and evolve. In addition, it needs to be of such quality that the organization's products and services are successful and result in the organization having sustained success.

Additionally, to have an SEI CMM Defined Process you need to not only establish the process descriptions necessary to assess at SEI Level 3 but need to have the integrated process trained, practiced, enforced, measured, and able to be improved through an institutionalized organizational process focus (Paulk et al. 1993).

Achieving a successful project- or organization-wide defined process is not something easily or quickly accomplished. Typically, significant organizational and cultural change is involved. This section covers a number of the key issues in achieving organizational and engineering success. These include understanding change, introducing process definition into your organization, the scope of process definition, and the use of metrics.

2.1 CHANGE

Software producers are affected by rapidly changing markets; tight budgets; increasingly critical roles for software in competences and products; new customer and standards requirements on software suppliers; and new, expensive technology. Your organization may be motivated to define and improve its process due to a variety of forces impacting software development and maintenance. Additional forces include: international competition, ISO 9000 compliance, the SEI levels of software process maturity, down-sizing of computers and staff, corporate TQM programs, and the use of software and computing in strategic, new, competitive products and services. This results in a need for more powerful, reliable, flexible, standards-compliant, and less labor-intensive approaches to software engineering.

2.1.1 ORGANIZATIONAL CHANGE

Organizations differ widely in the ways they address process improvement: some handle it as stressful episodes; others try to learn from each improvement but are poor at applying any lessons; and others treat process definition and improvement as a "core competency" with organizational structures, processes, and incentives aimed at maximizing their competitive advantage deriving from this area.

From this perspective, process improvement involves the managed exchange of information and resources both within your organization and with related, external organizations. Figure 2-1 shows a number of organizations and conditions that affect process improvement as well as the subsystems mentioned.

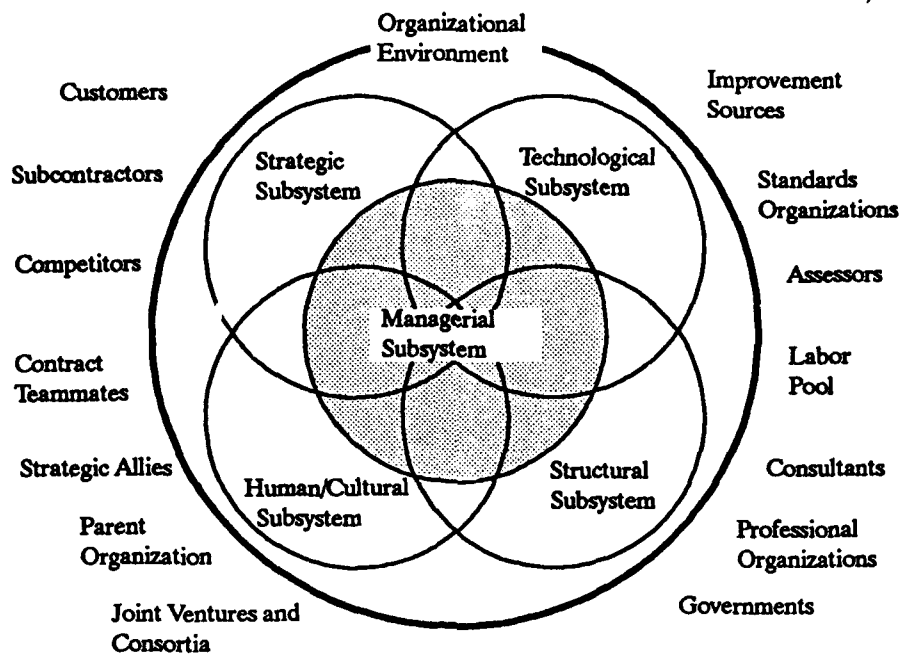


Figure 2-1. Organizational Subsystems and Context

When an organization views process improvement as an ongoing function, then organizational arrangements to handle technologies and process improvements may be instituted and improved. Each of these issues must be considered:

- **Strategic Subsystem.** Does your organization have a business strategy or do they simply react to whatever changes come along?
- **Technological Subsystem.** Are the processes used to transform inputs into outputs standardized and institutionalized? Do the processes rigidify operations or are they flexible? What types of technologies are being used?
- **Human/Cultural Subsystem.** What are the core values, behaviors, and “unwritten rules” that shape your organization’s culture? What orientations do people bring to work? Are they searching for challenge and involvement or simply working for money?
- **Structural Subsystem.** Is your organization bureaucratic and hierarchical or matrix in structure? Are Integrated Product Teams used?
- **Managerial Subsystem.** Does the dominant managerial philosophy stress accountability and control (authoritarian) or encourage initiative and enterprise (democratic)? Does the organization stress innovation and risk-taking?
- **External Entities and Relationships.** Do your processes mesh smoothly with your suppliers and customers? How do you assess and address customer satisfaction? Do you have mutually beneficial long-term relationships with technology sources? How do you compare to your current and future competitors? Do you exploit consortia?

2.1.2 ORGANIZING FOR SOFTWARE PROCESS IMPROVEMENT

A typical organizational structure recommended for software process improvement has a Software Steering Committee that coordinate, support, and guide one or more SEPGs that, in turn, create and support Process Action Teams (PATs) that do various process improvement tasks and participate in piloting new methodologies, techniques, and tools. This example structure will vary between organizations, but the roles enumerated in Section 2.1.3 will likely be common regardless of your overall organizational structure.

2.1.3 ROLES IN ORGANIZATIONAL CHANGE

Staff at any level in your organization can play one or more of the following roles in process improvement:

- **Sponsor.** This person possesses sufficient authority or influence to either initiate resource commitment for process improvement (authorizing sponsor) or reinforce process improvement efforts at the local level (reinforcing sponsor). Both authorizing and reinforcing sponsors continually legitimize and demonstrate ownership and commitment to process improvement. The departure or unavailability of sponsors could jeopardize the success of an improvement activity or group.

The authorizing sponsor is usually the senior manager of the organization and often serves as the chairperson of the Software Steering Committee. Reinforcing sponsors are typically at a middle manager level and are present within the Software Steering Committee.

- **Change Agent.** This person or team is empowered by sponsors to implement and facilitate process improvement throughout the organization. The SEPG and the PATs are considered change agents.
- **Champion.** This person advocates and publicly supports software process improvement in the organization, but lacks the power to sanction it. A champion can be present at any and all levels of an organization; successful ones are usually respected for personal or technical leadership.
- **Process Users.** This group of people implements the change and is the focus of the effort; i.e., they are expected to follow the process definition in the way they work, and are therefore likely to change their behavior. Throughout the cycle of process improvement, everyone affected by the change will, at some time, be changing some aspect of the way they work. Typically, process users are the people who develop your organization's software products and are commonly considered the technical staff.

Remember that any of these roles may, at some time, be subject to change (changing some aspect of the way they work). Sponsors may need to change their communication style to better stress the importance of process improvement. Change agents may need to develop their interpersonal skills to build strong teams within the organization.

These roles will evolve and overlap during your process improvement program. For example, a sponsor may start out as a subject of the change effort. The champion influences this manager to become an authorizing sponsor. Upon authorizing the proposed change, the sponsor may champion the improvement on a larger scale and to other organizations.

2.1.4 EVOLUTIONARY CHANGE PROCESS

To be consistently successful, process definition efforts need to be part of a viable and well planned and implemented process improvement effort. *Managing Process Improvement: A Guidebook for Implementing Change* (Software Productivity Consortium 1993a) gives extensive guidance on establishing a viable, well-managed process improvement effort. It provides a cyclic process for process improvement. Though process improvement activities and their dates cannot be accurately predicted in detail far in advance, *Managing Process Improvement* shows how a well-structured and executable plan can be developed by considering a set of core activities you can plan and execute in an iterative manner. Figure 2-2 presents an overview of this process:

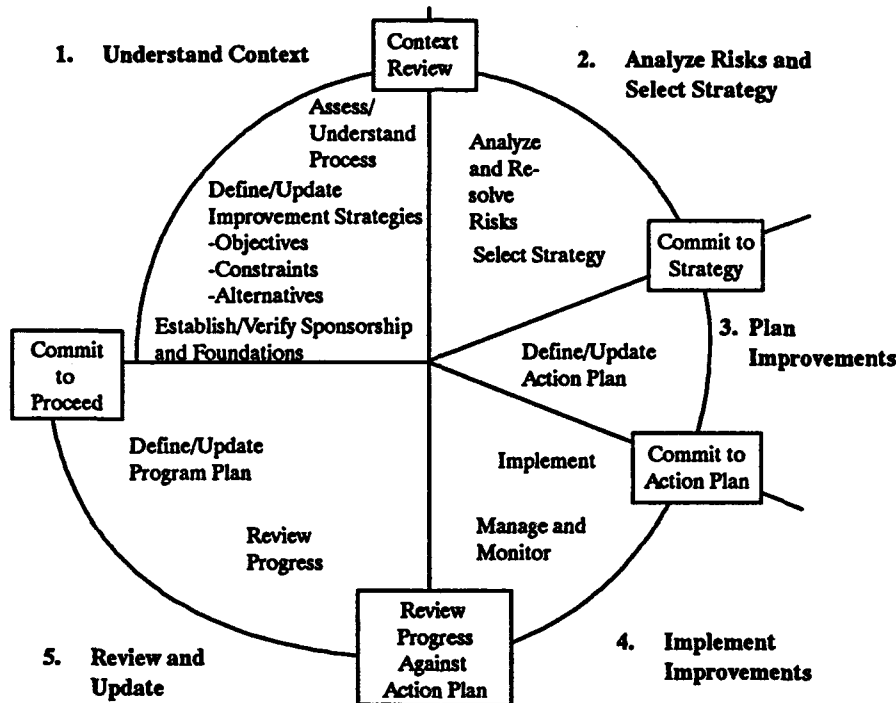


Figure 2-2. Process Improvement Process

The key steps are:

- **Understand Context.** Understand the current context including support for improvement, viable alternative strategies, and process.
- **Analyze Risks and Select Strategy.** Analyze risks and select a strategy for implementation.
- **Plan Improvements.** Plan the improvements for this cycle.
- **Implement Improvements.** Implement the improvements according to the plan.
- **Review and Update.** Review progress and update overall plans.

Advocating, planning, performing, reviewing, and managing process engineering and process definition are involved in potentially all of these activities, but the process definition activities concentrated upon elsewhere in this guidebook fit mainly into the Implement activity as part of carrying out an action plan.

2.1.5 STARTING WITH THE EVOLUTIONARY SPIRAL PROCESS

The Evolutionary Spiral Process (ESP) provides a starting place for a sophisticated, modern process. You should find it useful particularly if you have requirements for a substantial amount of variation and tailoring. It also is a good source of risk management methods if you want only them. It also contains activity definition outlines and artifact and role lists that may be useful as a starting point.

To date, most process models have been tied to a specific life-cycle model to the point that it is very difficult to distinguish the process from the life cycle. Another problem to completely tying the life-cycle perspective to process models is that life-cycle representations are linear and do not highlight the impact of feedback and iteration. A third problem arises when projects do not define a process model that would best suit their unique situation but rather attempt to use a model that is mismatched to project objectives and constraints because it is readily available or required by the customer. On the other hand, it is often difficult and overly constraining to define a model that is best suited to the project's objectives and constraints at the beginning of a project without knowing what surprises are in store as the project progresses.

The ESP model attempts to resolve these difficulties by providing a process model that:

- Is intentionally not tied to a specific life cycle
- Can be used to define and evolve a life cycle that will best meet project-unique objectives and constraints
- Will help incrementally generate the development process used to reach the life-cycle states

The ESP model is based on the spiral model, which was originally developed by Boehm (1986, 1988) to address the known problems in more conventional, primarily waterfall, life-cycle models. The spiral model explicitly incorporates the management steps missing from most of the earlier models. It permits its users to avoid the automatic adoption of one-size-fits-all processes and eliminates the lock-step or linear progression of stages characteristic of the earlier models.

The conceptual ESP model, shown in Figure 2-3, is essentially a management model. It is described by five main steps and several specific product and process management activities that can be used in conjunction with any life-cycle model. A cycle starts in the upper left quadrant with the step to determine objectives, alternatives, and constraints, then moves clockwise. In addition, you can follow the ESP model activities to determine a life-cycle alternative that will adequately address your objectives and constraints, incorporate the life cycle into the model, and subsequently evolve the life cycle as product development proceeds and objectives and constraints change.

The ESP model is meant to be repeated using the knowledge gained and lessons learned from the previous cycles; that is, you traverse all five steps of the ESP model one cycle at a time. A cycle is a complete traversal of all five steps that, when completed, matures the product by the amount defined in cycle-specific objectives and success criteria. A spiral is one or more cycles that, when combined, accomplish a specific objective, such as complete a project, product, work product, or other major milestone. A spiral may represent the complete life cycle or may include only the activities necessary to meet one or more of the life-cycle states.

By identifying and incorporating a specific life-cycle model into the conceptual ESP model, you can engineer a process model specific to your situation. In cases where the problem is well understood,

you may be able to successfully apply the waterfall, or top-down, approach that elaborates all requirements before starting design, completes the design before writing any code, and finishes the code before integration and test.

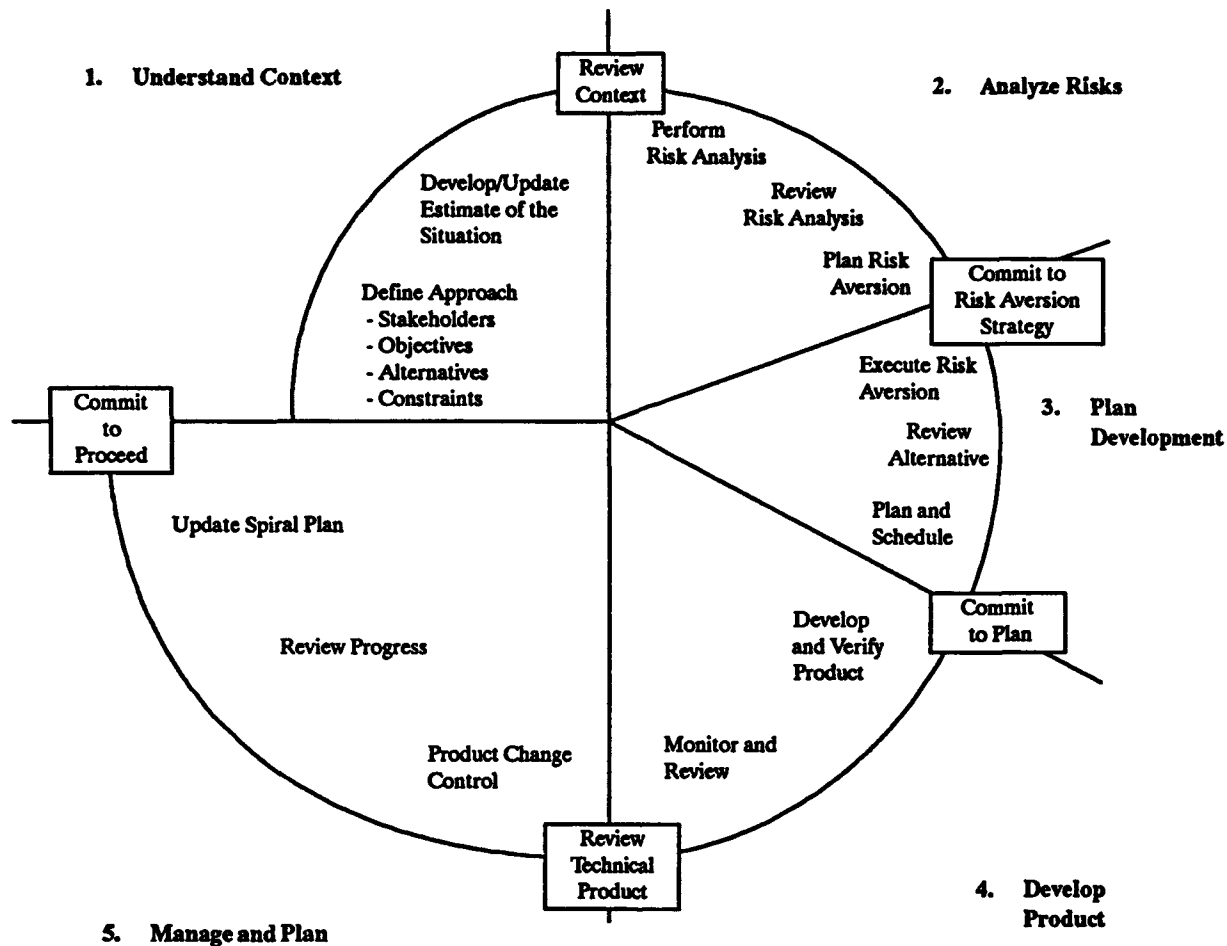


Figure 2-3. The Conceptual Evolutionary Spiral Process Model: A Management Focus

However, you may not have enough details to write the correct requirements until you have developed a prototype or performed some preliminary design. You almost guarantee some amount of rework if you adhere strictly to the top-down sequence. In this case, you may engineer a process model, such as the one shown in Figure 2-4, that permits a more opportunistic schedule for product development because it allows you to postpone some key requirements decisions where you might otherwise have been forced to make a choice with insufficient knowledge.

In the sample ESP model shown in Figure 2-4, you could have defined the life cycle in the first cycle and then updated the model to show the specific product life-cycle states and supporting activities in Cycles 2 through 4. As the spiral progresses, you can evolve the process model to accommodate changing objectives and constraints, identified risks, and lessons learned.

This example is only one possible elaboration of the spiral. Each project can have a customized spiral(s) to fit its situation.

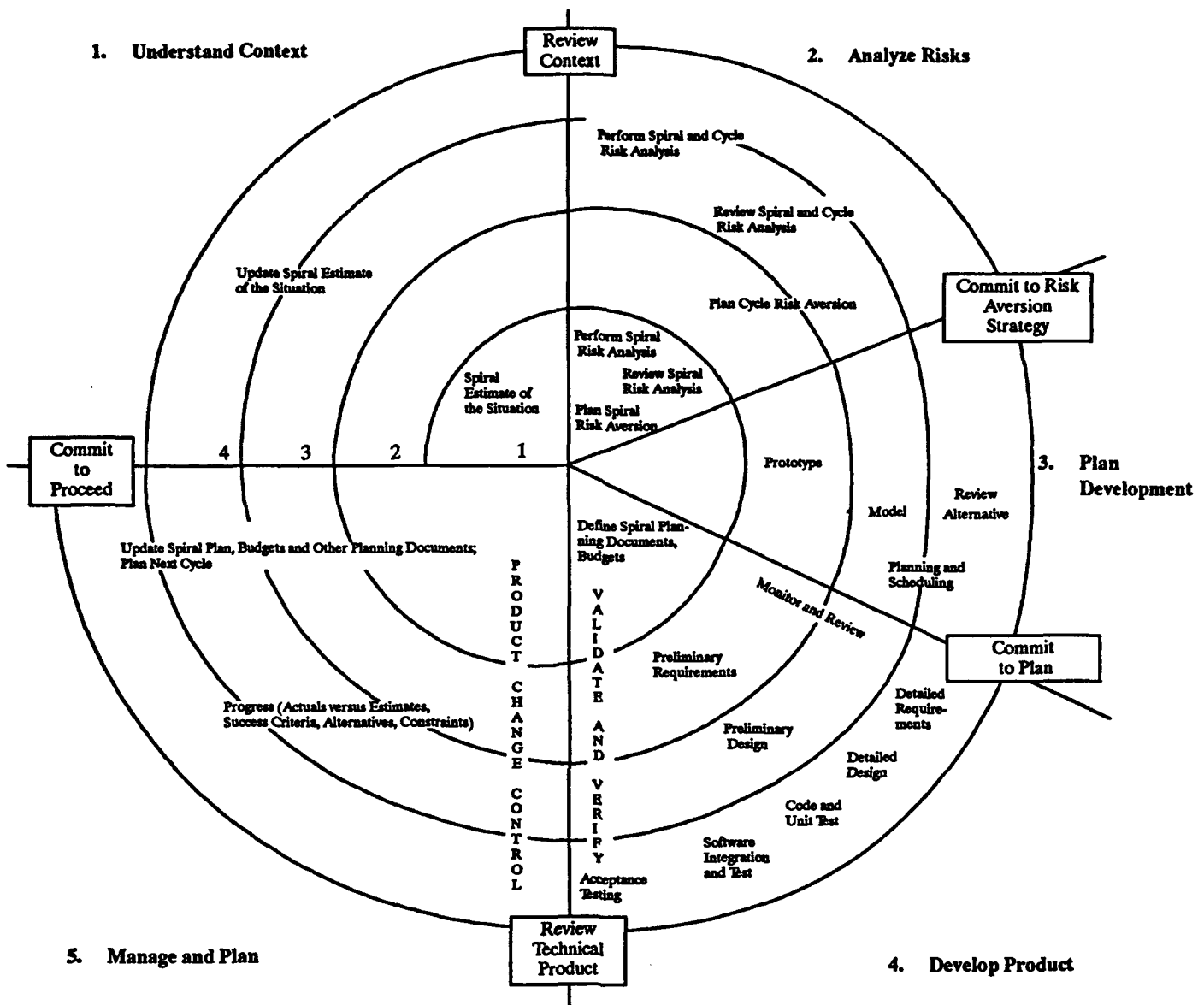


Figure 2-4. An Elaborated Evolutionary Spiral Process Model: A Complete Process Focus

Regardless of whether you are defining project level processes or organizational level processes, an iterative, spiral approach directly facilitates building and validating process descriptions that progressively improve in terms of depth, breadth, and maturity.

2.2 INTRODUCING PROCESS DEFINITION INTO YOUR ORGANIZATION

Successful process definition transfer depends on getting people to change—or, at least, formally document and control changes in—the way they work. Usually, you cannot be successful at getting people to change just by mandating it or by handing them a new process definition and expecting them to use it. You have to show them, in detail, how the change will impact their work in the short and long term, ensure that they are motivated to change, and then support them throughout the transfer. Figure 2-5 lists success factors that, if implemented, will improve your chances of successfully transferring a process definition into use.

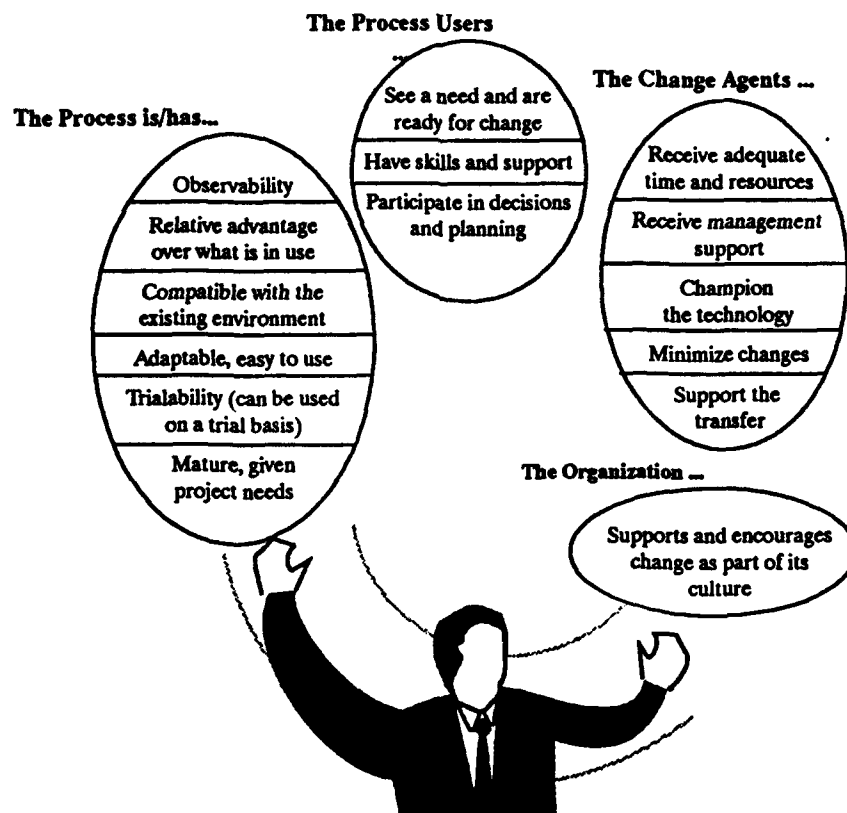


Figure 2-5. Success Factors in Process Transfer

2.2.1 TECHNOLOGY TRANSFER

The key challenge in process definition is not in defining processes, but in getting people to accept and comply with new processes. Technology transfer, or the insertion of new processes, is a field unto itself and is beyond the scope of this guidebook; further information is available in *Using New Technologies: A Technology Transfer Guidebook* (Software Productivity Consortium 1993b). Generally, however, you should consider the following guidance when starting or performing technology transfer:

- **Needed Commitment.** Your sponsors need to understand that if they attempt a transfer and fail, they incur the following costs (adapted from Implementation Management Associates 1992):
 - **Short-Term Costs.** Direct costs include wasted resources and failure to achieve a stated business objective. Indirectly, staff morale suffers because they may have invested their own time and energy into a failed transfer.
 - **Long-Term Costs.** If management commits to something that fails, then the staff will potentially have a lower confidence in management's leadership ability and will increase their resistance to the next transfer that they are asked to support.

So as not to incur these costs, your sponsors need to make sure they support the transfer throughout the entire process. They cannot just give the transfer lip service, expect it to succeed, and not expect any repercussions when it fails. If there is any doubt about whether you will receive ongoing support, then you should get new sponsors, delay the transfer until the support is received, or fail.

- **Scope.** Establishing the scope of the process definition effort needs careful consideration, particularly at the beginning. Among the factors to consider are your purposes (goals, objectives, motivations), constraints, alternatives, risks, opportunities, existing situation and plans, time frame, cost/benefits, and the exact products. In addition to organizational scope and users, you need to address depth, generality, adaptability, and integration.
- **Cost and Schedule.** Most technology transfers are expensive. If you are tight on resources, then you can possibly make up for it in the beginning by stretching out your schedule, but there will come a time when management will have to provide adequate resources. If you are limited on time and resources (and cannot change the situation or scale back to fit), the best guidance is to delay the transfer until your situation improves.
- **Timing.** Organizations perform work and for the most part can only perform work when they have a stable environment. If at all possible, time the transfer to occur at the beginning of an effort or at a major transition point when people are able to absorb changes better. It is nearly impossible to get an ongoing project to change, especially when it is in the middle of schedule and resource crunches.

As mentioned above, process improvement and therefore definition and transfer of processes into use is best done by a series of cycles rather than everything, everywhere at once.

- **Activity Duration.** Perform the transfer activities as rapidly as possible by performing activities and tasks concurrently. This might mean trading detail, elegance, and perfection (but not effective usability) for speed.
- **Cycle Duration.** The length of a cycle will be dependent on your situation. However, you can use the following rules of thumb when planning a cycle: the more people involved in the current cycle, the shorter the cycle should be because of the higher probability that risks will occur; and, the more comfortable your organization is with the activities planned for the cycle (e.g., training), the longer the cycle can be because your organization already knows how to mitigate most of the risks. One possibility is to time the cycles to correlate to your reporting cycle; for example, if management wants to see a progress report every 3 months, then you can time your cycles to be approximately 3 months in length.
- **Resources.** In general, transferring in a tool alone will take the least amount of resources, transferring in a method alone will take more resources, transferring in a tool and associated method together will take even more resources, and transferring in a system or software development life-cycle process will take the most resources. This is due to the increasing levels of impact on existing practices of each technology type.

2.2.2 UNDERSTAND YOUR PURPOSES AND THEIR IMPLICATIONS

Like any systems or software engineering effort, many of the potentially serious problems in achieving a defined process are related to requirements. What are the goals and their relative importance, who are the stakeholders and what are their concerns, what are the success criteria/measures, do the resources fit the requirements, and what requirements have the greatest risks are all typical questions that are important here as well. Your requirements for the process definition-related efforts exist in a context of organizational requirements.

Achieving a clear, shared, explicit statement of the objectives that reflect your needs in a particular situation is a key to success. Objectives can involve understanding (particularly understanding the requirements a process must meet), having an as-is definition, or improving. Process improvement efforts vary in scope and goals.

Scope of organizational objectives can be of five kinds (adapted from Venkatraman [1991]): (1) localized improvements, (2) improvement integrated internally across organization, (3) business process redesign, (4) business network redesign including suppliers, customers, and others; and (5) business scope redefinition where new process and technology are used to break out and change the organization's mission, scope, markets, and products. Your initial definition effort is more likely to be successful the earlier it is on this list of kinds.

If your situation is typical, it will be somewhat mixed with objectives varying for different parts of the definition(s). For some part, say configuration management, you may be satisfied with how this is done in a portion of your organization and want to capture these satisfactory internal example processes that you will then propagate. For other parts, your management's instructions may be, "We are lousy at that. Do not study how we do it now. Define us a good process." Of course, the greater the fraction that falls in the first category the easier introduction should be.

Identify the symptoms that your sponsor, users, and others most want to see change. Project overruns, shipped defects, excessive integration costs, time-to-market, SEI CMM level—what is it that will make it a success in their eyes? You will probably need to do some root cause analysis to go from symptoms to process causes.

Process definition can be used to help many kinds of activities. Figure 2-6 lists many uses of definitions. As shown, process definitions are used for anything from marketing, through process engineering and organizational change, to ISO 9000 certification. You should use this list of "process definition uses" to remind you of the uses that are important to you.

Marketing	Understanding	Institutionalization	Productivity
Transferring	Design	Cultural Change	Predictability
Selection	Definition	TQM	Time-to-Market
Trial	Analysis	Process Improvement	Cost Lowering
Tailoring	Reasoning	Process Reengineering	Quality Improvement
Training	Review	Tool Development	Standardization
Learning	Integration	Environment Development	Vendor Evaluation
Remembering	Change	Team Building	SEI CMM Assessment
Performance	Reuse	Organizational Unity	Certification (e.g., ISO 9000)
Planning	Traceability	Multiorganization Coordination	
Measurement	Communication		
Management	Knowledge Capture		
Quality Assurance	Shared View		

Figure 2-6. Uses of Process Definitions

Initially, much of your process definition effort is likely to be concentrated on capturing descriptions of existing processes for understanding requirements and as input into initial definitions. The common process for discovering and capturing these processes resembles the processes long used to analyze processes in systems, software, and industrial engineering. (See Sections 2 and 4 of Volume 1.)

Thus, establishing the scope of the process definition effort needs careful consideration, particularly at the beginning. Among the factors to consider are your purposes (goals, objectives), constraints, alternatives, risks, opportunities, existing situation and plans, time frame, cost/benefits, and the exact products. You should outline the definite inclusions, commonalities, variabilities, and exclusions—what is new and what unchanged. You also need to address depth, generality, adaptability, integration, and change over time. You need to verify your initial statement of scope against process improvement plans and validate it with all affected parties.

2.2.3 PROCESS DEFINITION AND MODELING STAFF

When introducing or expanding a process definition and modeling effort, you must determine which roles are necessary for supporting the work and what types of skills are desirable for people performing those roles. Arguably, three to five matrixed engineers and one matrixed manager are necessary for establishing a process representation (definition and modeling) program. Because some training is involved, there is too much risk in training only one person and then having the entire program depend on the efforts and availability of that person. In principle, it is better to have more people contributing smaller amounts of their time than to have fewer people contributing significant amounts of time. Once a program is well underway, it is ideal to have one or more people work their way into full-time responsibilities as “process engineers.”

The preferred candidates for such work are those that already have a process-oriented background. This group includes people who have had experience in systems analysis and design. These individuals have had considerable exposure to **process control** at a systemic level. Also included are those individuals who may have received exposure to systemic process issues through involvement with TQM, process assessment efforts, and similar activities.

The learning curve involved in becoming proficient in MPDM is comparatively nominal, so the benefits of having several people trained and potentially available (as matrixed resources) for process representation efforts is relatively low-cost and low-risk. This is especially true when you consider the value derived from the increased variety of insights and experience.

As discussed in Section 5, the templates can be used as a point of departure for alternative process representation notations such as ETVX, SADT, and Petri Nets. If such notations are one of the eventual goals of the process representation effort, it is highly advantageous to select process engineers partially as a function of whether they have a background in the alternative notation of choice. In all cases, using alternative notations for constructing process models is consistent with their use in constructing software models. Consequently, notation-specific skills previously acquired by process engineers can be applied to process representations using those notations.

2.2.4 ORGANIZING FOR PROCESS DEFINITION ACTIVITIES

It is suggested that process definition, as an organizational process, be the first representation constructed. Then, when performing the subsequent process representation effort, you will be following the model constructed during your first effort.

Much is said elsewhere in this guidebook about steps and practices for creating process definitions. This subsection discusses related activities that any engineering efforts most also address.

2.2.4.1 Creating Process Definition

If you have a limited project or program scope, then the material in Volume 1 suffices. If you have organization-wide requirements or those involving substantial variations and tailoring, then the

material in Volume 1, combined with the the Tier 2 fields discussed in Section 4, will suffice. If you plan on process simulation and enactment, then the material in Section 3, and the Tier 3 and 4 fields presented in Section 4 will be helpful.

2.2.4.2 Verification and Validation

Verification and validation of process definitions occur repeatedly. You verify a definition by investigating whether the process is being defined in a manner consistent with requirements; you validate a definition by investigating whether the right process is being defined. Reviews and inspections are the most common techniques for verifying and validating process definitions, but pilot trials, simulations, and analytical modeling are also possible.

Validation and buy-in are important to the success of process definition efforts. These should start at the beginning and continue throughout the entire effort.

2.2.4.3 Process Definition Management

Like any engineering endeavor, the process definition efforts need to be managed. Commonly, this will be done in the context of process improvement and process engineering. All the principles of good engineering management apply—particularly, concern for user involvement and satisfaction. You need to give attention to planning process definition releases and transfers/installations, installing processes, and providing operational support. (Process definition management issues are discussed in detail in Section 2.2.7, and throughout Section 3.)

2.2.4.4 Process Definition “Process” Management

The process definition process needs all the normal infrastructure of an engineering endeavor including quality management, process definition improvement, configuration management, documentation management, process definition support, “system” administration, and training. These need not be elaborate or require separate organizational entities. You may want to take advantage of existing infrastructure to accomplish many of these activities.

Not only should the definitions continually improve, but the process that is used for process definition should also be systematically improved. This can help set an example for software process improvement.

2.2.5 GETTING STARTED

Once you have identified who will be working as process engineers, the next step is to identify the first area to be defined. Your own process definition process can be a good example to learn on, but do not spend excessive resources or unreasonably delay the main work.

Before actually performing process representation, you must train the process engineers and, if possible, those who will be managing the efforts. MPDM is not complex; therefore, training can be limited to approximately 2 days. However, this depends on requisite background and related topics also introduced during training, such as metrics, process maturity levels, and transferring processes.

For further information on getting started, see Volume 1, Sections 2 and 4.

Once you have a sufficiently complete process representation, it can be analyzed for areas of risk and for opportunities of improvement. Often, simply having an entire process detailed in an easily accessible and understandable format allows you to think about and discover risks and opportunities. For example, you can examine the templates with regard to the average number of entry criteria and average number of exit criteria specified for events. Then, you can reevaluate events whose criteria significantly exceed the average to determine whether such excessive binding to other events constitutes unusually high risk.

The eventual goal of analyzing models for risks and opportunities is to derive new and improved models. You can then carefully analyze the new models for consistency, completeness, and potential problems. When management is confident that the proposed process is well understood and has sufficient potential benefits, they may decide to institute an exploratory or pilot project based upon the proposed process.

2.2.6 ONGOING IMPROVEMENTS

Improvements can occur in both the processes and how they are represented.

2.2.6.1 Process Improvement

Process improvement stems from using the baselined process and evaluating how well it works, and from evaluating and monitoring the product being produced as a result of enacting the project process. Process evaluation techniques can include internal and external process assessments, ongoing measurements and metrics, and post-mortem evaluations.

An internal process assessment assists an organization to determine and characterize its internal process maturity, mobilize support, and guide improvement. An external measure of organizational process maturity can be used as a way for a customer to assess the software development capability of potential contractors or vendors.

Cost, schedule, size, error, and risk measurements are all very useful project management data. They can be collected in the process measures database and used to build process metrics, such as the average percentage of cost and time spent at each activity, or the identification of the activities where errors and risks are typically discovered or introduced.

A post-mortem evaluation of a project's process model and enactment might also identify potential reusable process assets such as process(es), methods, tools, and measurements, as well as the rules for reusing these process assets. Process evaluation can include classifying and analyzing the different development process data by business areas to leverage similar process metrics of cost, time, errors, and risks.

Both originally and as part of evolution, you need to give particular attention to planning process definition releases and installations, installing processes, providing operational support, and identifying new operational capabilities.

2.2.6.2 Representation Improvement

Once you have used the templates to construct one or more process models, you will have insight into how the templates can be improved and explicitly tailored toward your needs and the nature of the models being constructed. You can add new fields, develop new classes of templates, include new

meta-classes, and introduce even further levels of abstraction. In all cases, it is important to monitor the benefits when contrasted with the additional effort and complexity introduced by such extensions.

The recommended approach is, of course, to automate support for process representation and to work with electronic versions of the templates. This not only allows for virtually unlimited field sizes (constrained only by limitations in a supporting database, for instance), it also introduces the possibility of active links between the templates. Relatively trivial automated support allows you to easily follow relationships between templates. This greatly improves the ease with which templates can be improved.

Once you are familiar with the development and use of process representations within your organization, you can look for ways in which process modeling can support other organizational endeavors, and vice versa. One major candidate for potential mutual support is with organizational metrics and measures. Although many organizations have established measurement programs without using process representations or models, you should note that these programs can readily facilitate each other. Hence, you may want to tailor your templates to directly support a program of process measurement. As discussed in Section 2.4, metrics are an excellent means to gain insights into dynamic process characteristics.

2.2.7 GUIDELINES FOR MANAGING PROCESS DEFINITION

Managing the process definition effort involves the full range of management issues. Although many of the points made elsewhere in this section have management components, the guidelines here directly address leaders of process definition efforts.

Process definition should be an example of a well-run effort.

1. Set explicit, realistic goals—both near- and long-term.
2. Adopt desired process behavior. While one cannot reasonably expect your software process definition effort to be much better organized and executed than your software projects, you should strive to adopt the next round of improvements desired of such projects. For example, if you expect software projects to use sound estimating procedures, then the process improvement program should use them. If you are aiming toward having a defined software process, then the process definition effort should define its process first.
3. Use the eleven-step scale (shown in Figure 2-7) to determine local cultural norms in handling risk or need to change (Grove 1983; Charette 1992; Redwine and Riddle 1985; Sage 1993). Build a culture within the process definition effort (and hopefully the whole process improvement program) that is climbing the eleven-step scale (Figure 2-7) ahead of the rest of the organization.
4. Explicitly manage uncertainty and risks. Be particularly sensitive to the need to provide your sponsors with near-term, tangible successes. Though process improvement economics is in its infancy, you should strive to justify process improvement efforts from the perspective of “return on investment.”
5. Have your own explicit process improvement effort.

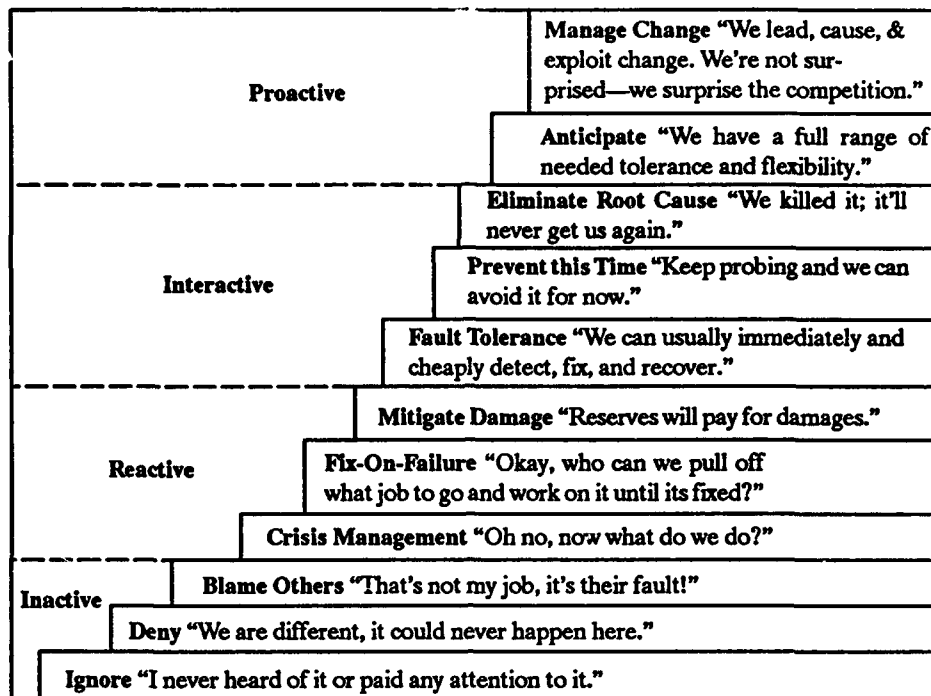


Figure 2-7. Eleven-Step Scale for How Organizations React to Risks or Need to Improve

Respond to your customers.

1. Listen to process users and sponsors (e.g., Steering Committee). Follow the motivations, the threat or risk vectors, the money, the process users' needs and their customers' needs in deciding what to do. Motivated process definition users are the keys to success.
2. Be aware that the list of stakeholders may change. Try to obtain support broad enough to include likely new stakeholders (successors). This affects the amount of time and resources needed to get commitment.
3. Increase sponsor and management commitment and enthusiasm by building increasingly open, ongoing relationships with more key persons, and ensure that they see you as yielding solutions, not surprises.
4. Rotate process users through the process definition effort.
5. Have validated requirements for your process definition efforts.
6. Measure diffusion, success, and satisfaction.

Integrate the process definition effort with organization processes.

1. As far as possible, involve yourself in all related planning (including budgeting) and with similar initiatives.
2. Increase awareness, coordination, and integration across software process improvement and definition efforts throughout the organization.
3. Ensure that the process definition effort's own processes integrate smoothly with other organizational processes.

2.2.8 PROCESS ENGINEERING

This section covers issues such as: Are the processes used to transform inputs into outputs standardized and institutionalized? Do the processes rigidify operations or are they flexible? What types of technologies are being used?

Institutionalization is the extent to which the use of the process is routine, widespread, and embodied in the organization's governmental mechanisms. Only a summary treatment is provided here; for further information, see *Managing Process Improvement: A Guidebook for Implementing Change* (Software Productivity Consortium 1993a) and *Process Engineering With the Evolutionary Spiral Process Model* (Software Productivity Consortium 1994).

In your process engineering, use the best practices and ideas from systems engineering, organizational development, software process improvement, user, and other communities.

1. Support learning. Train and educate your process definition management and staff in new software process engineering-related skills and knowledge. Most software process knowledge and skill has an even shorter half-life than regular computing knowledge; this requires continual study, training, and education.
2. Copy the best involvement mechanisms in use in your organization. For example, joint application design, integrated product/process teams, or concurrent engineering teams.
3. Pay attention to process requirements and integration.
4. Use process (re)design expertise to improve processes' speed, dependability, cost, and resulting customer satisfaction.
5. Cultivate the persons performing the process to become your biggest source for improvement suggestions; then use them.

Stay on top of new process developments.

1. Access and digest relevant periodicals, market reports, and government studies. Subscribe to services, databases, and analysis groups.
2. Use existing data and knowledge wherever possible. In technology areas that change rapidly, be careful of data over 2 to 3 years old.
3. Use traditional and nontraditional sources, as well as local and global sources, for new processes and technology. Traditional sources include consulting firms, colleagues, literature, seminars, internal R&D laboratories, universities, computer manufacturers, software tool vendors, and meetings. Less traditional but sometimes even more effective sources include joint ventures, consortium membership, federally funded R&D centers, competitors, suppliers, and customers.
4. Participate in or track appropriate standards projects, e.g., SPICE.

Exploit automation.

1. Use office automation to manage and communicate.
2. Use automation to support process definition and modeling.
3. Generate an electronic "paper trail" on your analyses, decisions, rationale, and implementation efforts. To support this, many activities do not prescribe a specific documentation type (e.g., a formal plan). Type of document you create is up to you and your management.

Institutionalize the process definition effort.

1. Define the software process definition process and evolve the definition.
2. Achieve a workforce skilled in its use.
3. Embody the process in organizational governmental mechanisms such as policies, procedures, measurements, reports, reviews, assessments, and audits.
4. Incorporate representations in initial personnel indoctrination and training and other mechanisms of acculturation.
5. Reinforce the process to prevent your process improvements from slowly fading away.

2.2.9 EXTERNAL ENTITIES AND RELATIONSHIPS

For your organization to be a permanent leader in process definition, it will need to develop stable and long-term relationships with other important external organizations, including your customers, suppliers (e.g., vendors, consultants), and industrial peers.

Ensure strong and improving relationships with external organizations.

1. Encourage staff, user, supplier, and customer involvement in decisions that affect them.
2. Attempt to establish relationships so you can design processes appropriate for supplier-customer pairs or chains with your subcontractors and clients.
3. Form ongoing relationships with select process definition and technology sources. This can take forms such as industrial liaison relationships with universities, contracts, joint ventures, consortia, periodic benchmarking, strategic alliances, investment, or cross ownership. Ongoing relationships will help facilitate transfer, give you advance insight and influence on how new processes will fit into your environment, and promote ongoing support for the transfer from the source.
4. Provide a vehicle for systematically asking customers and suppliers about their immediate and expected future process and technology needs and changes.
5. Carefully select and form a relationship with assessor organization(s) and personnel that can come to know your organization and approach to improvement.

6. Find ways (e.g., industry surveys, benchmarking, visits, business intelligence) to compare your organization's software process and software process improvement program to your competitors.

2.3 THE SCOPE OF PROCESS DEFINITION

Particularly if you are leading or planning a process definition effort, you should be aware of the full range of activities and future possibilities involved—even if, as is likely, you concentrate on the definitions for immediate use. Process definitions can exist at different levels of generality at different levels of the organization (as discussed in Section 3.3), contain multiple parts, use a variety of representations, and change over time. Process definition efforts involve the full range of engineering activities as applied to process definitions, including verification and validation and configuration management of the process definitions themselves.

2.3.1 FORMS OF PROCESS DEFINITIONS

Process definitions describe some or all of their process's aspects—e.g., goals, activities, artifacts, ordering, organizational structure, resources—at varying levels of specificity and scopes of applicability. They are used, possibly in different forms, to constrain, engineer, and perform processes. Examples include corporate standards, guidebooks, plans for human use, and scripts for computer execution. The emphasis in this guidebook is on definitions that interact with humans for engineering, performance, or other purposes.

The most common technique for representing or defining processes is the use of descriptive text. Generally, operations manuals are text-based representations that describe one or more processes. Organizational policy and procedure manuals are also potential sources of process descriptions. One significant advantage to text-based descriptions is that there are virtually no constraints placed on the description itself. Because the notation is the entire language of words, anything that can be talked about can be described. This, however, is also the most significant disadvantage to text-based process descriptions. Because no constraints exist (other than grammar) on the structure of the description, the potential for inconsistency, ambiguity, uncertainty, and inaccuracy is considerable. Text-based descriptions are typically the least formal type of process representation.

At the other end of the formality spectrum is mathematics. Although mathematically provable process models are an ideal goal, such methodologies are still an area of active research, and there are not any clearly cost-effective and widely applicable math-based methodologies. However, between the informality of descriptive text and fully rigorous formal mathematical descriptions of all aspects, there are many options for representing processes at varying degrees of formality.

Having graphical support for a notation can directly contribute to having some degree of formality. Usually, a graphically-oriented notation comes coupled with a methodology that imposes rules on placement, use, and connections between the graphical objects. These rules constrain the types of structures that can be built and increase the formality of the resulting depictions. Additionally, graphical depictions are especially useful for portraying abstract or high-level relationships. The resulting diagrams allow for insights into the process that can be virtually impossible to derive from descriptive text.

2.3.2 ARCHITECTURE, ACTIVITIES, AND METHODS

To facilitate construction, evolution, reuse, and adaptation of software processes, software **process architectures** are useful just as they are in other types of systems (Redwine 1991). While architectures

are sometimes talked of as just the structure and relationship among the components of a system and the system's interfaces with its operational environment, architectures fulfill a number of purposes. For process architectures these purposes include: describing the significant components and structure; reuse variations; guiding tailoring (selection, adaptation, and composition), instantiation and binding; smoothly assembling the components; and gracefully evolving over time. An architecture must, of course, allow provision of the needed functionality and performance, but it should also, where possible, facilitate their provision and use. Architectural design decisions tend to be the earliest made and the longest lasting of process definition design decisions.

Architecturally identified activities can be black boxes, possibly arranged in levels of decomposition, to be filled by methods performed by tools and persons. Activity black boxes are generally described by their entry and exit criteria, including descriptions and constraints on input and output work products. The activity interface also needs to accommodate management direction as input and measurement and management data as output. These include progress/status reports, problem and risk reports, recommendations for higher level decisions, and process and product measurements including quality measures. Figure 2-8 shows the elements typically involved in performing an activity.

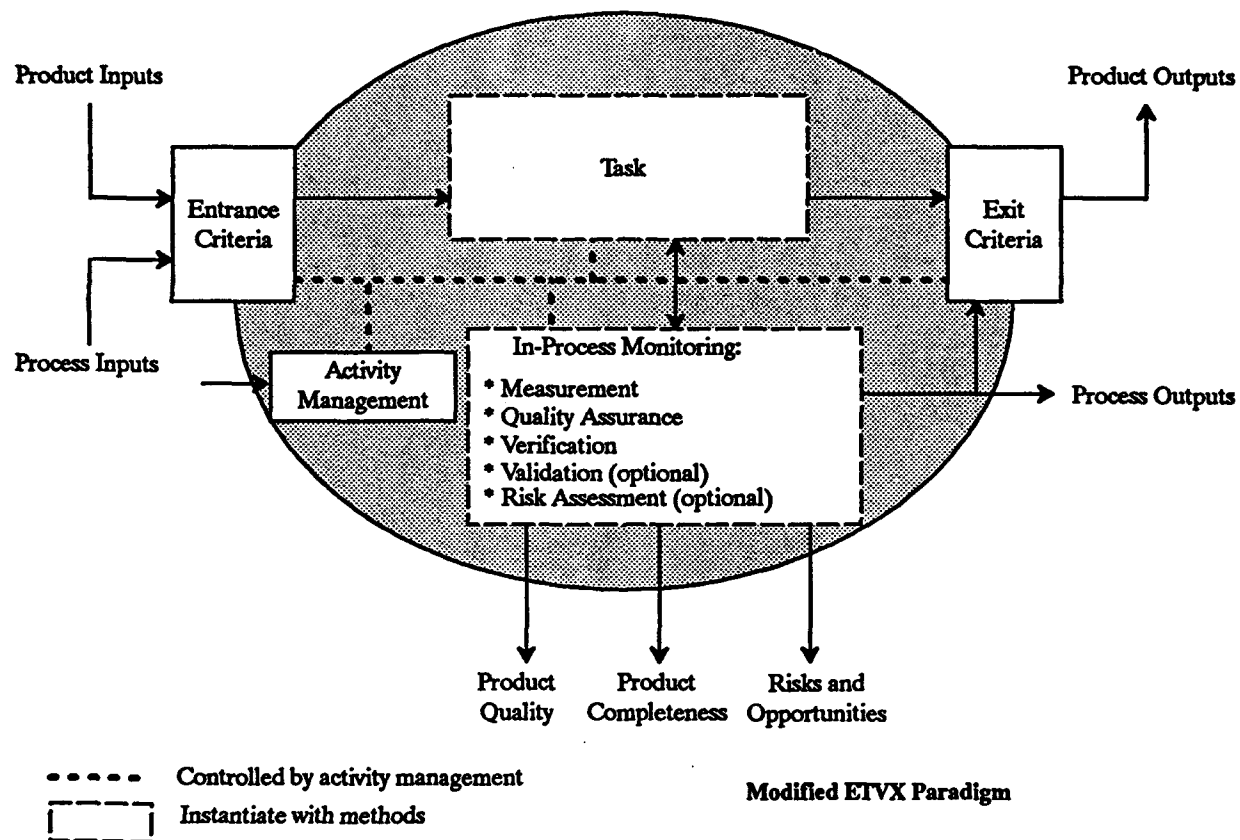


Figure 2-8. Typical Elements of an Activity

Methods fill in activities and may include or reflect details of artifacts produced. An architecture places constraints on methods used to accomplish activities. Methods may be required to provide data details, progress measurements, continuous-improvement measurements, and information for project planning and estimating. This is in addition to local management methods, appropriate verification and validation methods, and conformance to process integration standards. Finally, methods need to treat many tailoring/instantiation variables, such as the degrees of software systems'

decentralization and dependability criticality—a method may be appropriate only for a subset, adaptable across a range, or unaffected by variation in such an aspect or process definition “driver.”

2.3.3 EVOLUTION OF PROCESS DEFINITIONS

Definitions evolve to help improve the process in general and to respond to requirements and experience. Figure 2-9 shows some of the ways in which definitions evolve. At the project level, two versions of the process definition are shown:

- The As-Is Project Definition that is currently being enacted
- The To-Be Project Definition that is planned to replace the as-is definition

As-is definitions can be derived from tailoring program-level, business area, or organizational definitions, or from analyzing and capturing current practice. The latter is a common source when organizations are just beginning to define their processes. Typically, the best of these become candidates for generalization and inclusion in organizational-level definitions.

To-be definitions may evolve before implementation as the result of analyses (both static and through process simulation) or after implementation as the result of learning from experience. The project-level experience can also be a source of improvement for the organizational level. Results of enactment across projects can be measured and analyzed for such things as activities that originate the most serious defects, and improvement of these can be addressed at the program, business area, or even organizational level.

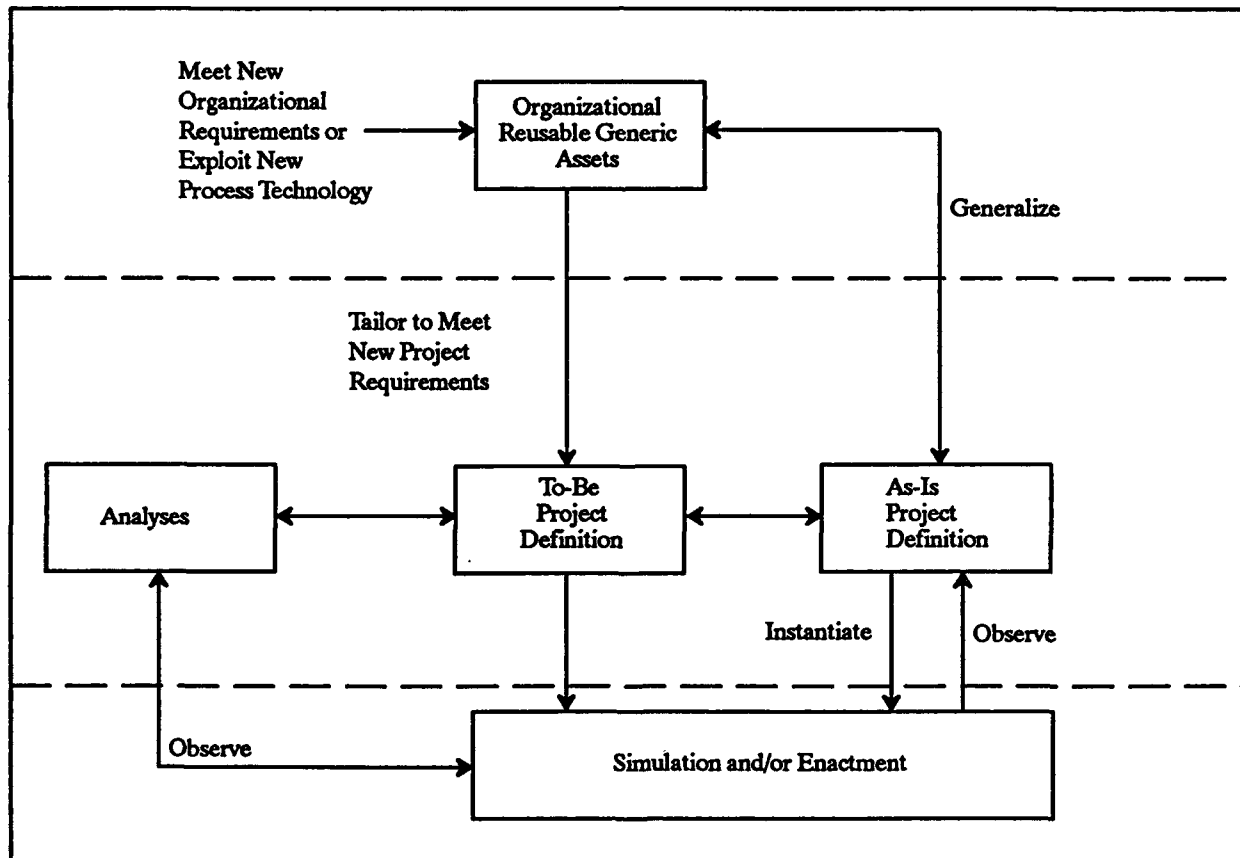


Figure 2-9. Process Definition Evolution

2.3.4 ADVANCED PROCESS DESCRIPTION USES

While this guidebook version gives them limited treatment, two of the areas with strong future potential for using process descriptions are automated support for processes based on their definitions and simulation of processes for analysis, planning, improvement, and training.

2.3.4.1 Automated Enactment

Over the last decade, a series of International Software Process Workshops has been held, mainly inspired by the potential for automated support. Although they are limited, early products are beginning to appear not only targeted to software process support but to office work-flow and business reengineering. Some groupware products have potential to provide some support to processes. Scripts or more advanced mechanisms compose and orchestrate the execution of multiple tools with little or no human action.

This is an area with considerable promise, but limited products. Nevertheless, awareness of these technologies may be important to your mid- and long-term planning.

2.3.4.2 Simulation

Simulation tools originally intended for other or general purpose uses have been used to try to simulate software processes. Experience is very limited but shows some promise for engineering, management, and training. The creation of "virtual projects" with stochastic simulation allows both analysis of what may happen and case studies for training. The analogy of flight simulation is used by some proponents. One advantage is that the effects of the stochastic and multiple feedback loop nature of projects and organizations are much easier to appreciate with some of these simulation techniques.

As in all modeling of human organizations, many questions exist as to the appropriate purpose, scope, and fidelity for a model to be useful. Definitions that can be modeled and instantiated to degrees needed for useful simulation will offer significant advantages in the long term. The Tier 3 and Tier 4 fields discussed in Section 4 support the extension of your process information toward the more rigorous descriptions needed for simulation and other automated analysis.

2.4 USING METRICS TO STEER TOWARD SUCCESS

The purpose of metrics is to increase objectivity by providing a common measure with which to compare different phenomena or the same phenomenon at different times. However, in addition to the considerable benefit derived from using metrics, there is also the risk of mismeasurement, or misinterpretation. Section 2.4.1 covers important characteristics of metrics: both from the perspective of benefits and from the perspective of associated risk.

2.4.1 IMPORTANT CHARACTERISTICS

There are many important aspects to metrics, but four of the most important are:

- ***Metrics must correlate to the phenomenon being measured.*** From one perspective, metrics are intended to facilitate insight into some phenomenon in the real world. The intent of the metric is to change in some manner that is consistent with changes in the phenomenon being measured. If a metric does not correlate to what it should measure, it is useless. The more closely changes in the value of a measurement parallel change in the characteristics being measured, the greater the usefulness of that metric.
- ***Metrics must measure something of interest.*** In addition to correlations, metrics must also measure something that is interesting. Interesting is invariably subjective, but the concept is crucial nevertheless. It is too easy to define a large collection of metrics and spend disproportionate amounts of time and effort measuring characteristics that yield little, if any, valuable insight. From this perspective, interesting metrics are those that provide you with insights that affect your decisions or your actions. Clearly, when you find that you largely ignore a particular metric and it does not influence your thinking, that metric has ceased to be interesting for you.
- ***Metrics must reflect something that can be influenced.*** Another key characteristic of metrics is that the measure relates to something that can be influenced. There is little value gained by measuring something that has no relation to anything that can be influenced. Curiosity may be satisfied, but if the metric monitors something completely outside your sphere of control and if nothing related to that metric can be influenced by you, the metric does not allow adjusting a process to achieve changes in the data reflected by the metric. If the metric can in no way provide you with any useful guidance on how to alter the phenomenon being measured, or your actions with respect to that phenomenon, then the metric is essentially meaningless.
- ***Metrics must support making corrections or predictions.*** Finally, it should be stressed that one of the primary benefits to metrics is their use in validating predictions. For example, when proposing changes to a process model, you should attempt to predict what effect the changes will have on the metrics monitoring that process. When proposing a new or modified process, such proposals should be accompanied by claims to the effect that metric-a will go up by x units and metric-b will be reduced by about y percent, etc. One value to such predictions is that they more precisely communicate the expected benefit from instituting a particular change.

Of even greater value is the fact that this approach can contribute to verifying whether the actual results of the new process match the predicted results. If results are generally as predicted, there can be a higher degree of confidence that the predicted benefits will likewise be manifested. If changes in the metrics are running opposite to their predicted direction, it may be time to consider reinstituting the original process and reexamining the new process.

Please note, however, that it is not always the case that when the predictions are wrong, the process is wrong. It could be that the metrics are not measuring what they seem to be measuring or that there are confounding influences affecting the measurement that are not being considered in the interpretation. Simply put, the problem may not be what is being examined, but the way it is being examined. Fortunately, using metrics to make predictions is a self-correcting problem. Either the metrics are high quality and your ability to make predictions improves, or the metrics are not high quality and the lack of predictability causes you to reexamine and improve the metrics. In any event, you gain insight and make progress.

2.4.2 INCORPORATING MEASUREMENT IN PROCESS DEFINITION

Considerations of what measures to include involve three factors: a minimum uniform set, measuring to meet your organizations goals, and a set of measures for each activity (and possibly each other type of item, e.g., artifact).

When establishing measures for an activity, a useful rule of thumb is to include measures of input size and quality, resource usage, and output size and quality. Note that this generally implies size and quality measures for products that could benefit from standardization across activities.

2.4.3 METRIC SUPPORT FOR PROCESS TRACKING AND COST MODELING

This section describes activity-based cost models and how you can apply them in starting your process modeling effort. To implement process models, you need to accumulate software process and product information. Many existing accounting systems maintain costs in terms of activity-based models. Yours should do so in terms of the activities in your process definition.

This section presents methods now used to work with the software development processes in your organization. The methods express each major process in terms of an activity. The methods are used by management for activity-based cost estimation. Activity-based models use a bottom-up approach to software development cost estimation based on an analysis of the costs of the individual activities that compose the software development process. Activity-based models are especially effective in an environment in which you have established a software experience database and where you use that database to feed back information about the process to improve the process.

The cost estimation methods usually base estimates on a function of the software product size. In turn, the development schedule is estimated as a function of the cost estimates. The size, cost, and schedule parameters are closely related.

The methods presented in this section roughly correspond to a progression through the SEI process maturity levels. If your software development organization is at process maturity Level 1, the initial level, you probably have no experience data in a database. By the time your software development organization is at SEI process maturity Level 2, repeatable, you will have established a software experience database and accumulated sufficient data so that you can calculate unit costs for the main activities of software development (e.g., requirements definition, design, code and test, integration and test) and apply them on a project basis.

An activity-based model is built by assembling and ordering the activities that compose the development process to be used to produce the intended software product. The activities that form your development process may be from a previously used process, or they may come from a modified version of a previous process with some activities removed and other activities added. Alternatively, they may be a selected subset of activities from a "menu" of activities. An activity-based model enables you to begin by using the resource consumption (cost) for each of the activities in the development cycle, such as requirements analysis, preliminary and detailed design, code and unit test, computer software component integration test, and computer software configuration item system test. Your project may not use all of the "repertoire" of possible activities. For example, if you are developing a new version of an existing system, you might not have any preliminary design in your development process. Define your software development cycle in terms of known and measurable activities based on your organization's experience as contained in your experience database.

Often, a computer-aided software engineering (CASE) tool is seen as a potential approach for reducing project cost. When considering the use of a CASE tool, you must determine (or estimate) its effect on specific activities in the software development process. This consideration should be tempered by a general recognition of the fact that some aspects of an activity are subject to automation while others can only be done by a person. For each activity, you should determine the impact of the CASE tool application on:

- The reduction in the unit cost of doing the activity
- The additional cost, if any, on this and any other activity of applying the CASE tool
- The inputs to and the outputs from the activity
- The determination of how and when the activity is completed
- The quality of the activity
- The sequence of activities

You must not only consider the potential impacts of the application of the CASE tool on each activity but also the possible interactions of the applications of several CASE tools. If investment (in the tools or the process) is the same for each application, you must recognize the possibility of a decreasing return on investment.

2.5 SUMMARY

This section covered variations among process definition and modeling situations and provided guidance you can use to improve your own process definition effort. Much of the guidance derives from seeing what others have done in the past—particularly what they have overlooked. However, you can certainly be successful without following every piece of advice, and this guidance should always be evaluated with respect to your particular situation.

Each time you want to improve your process, consider process definition an ongoing function with a permanent set of organizational processes and relationships. Initially, some of this may be a luxury in your situation, but you should revisit these aspects as you are considering your second or ongoing effort.

Process definition success depends on the same factors that make systems or software projects successful. Use your experience, involve your users and funders, set realistic objectives, manage your requirements, have competent staff, have good organizational relations, verify and validate throughout, and use good management and engineering practices. Remember your organization only benefits if people use the definitions to do better work.

As a word of caution, when process definition efforts fail, it is often due to either: (a) organizational, cultural, or management reasons, rather than technical; or (b) by taking on too much too soon, rather than taking an incremental approach, building incremental experience, and providing incremental benefit. The road to successful process definition is the same road that leads to success in other projects: plan, manage, do quality work, be efficient, involve your users, get early results, and demonstrate ongoing success to your sponsors.

3. ORGANIZATIONAL PROCESS DEFINITION

Recognizing that process definitions could exist at the organizational level and require tailoring to a variety of dissimilar projects, the previous sections of this guidebook addressed such situations, but only at an introductory level. This section provides further details on the activities of organizational process definitions. In part, it also provides the motivation for some of the more advanced template features described in Section 4. Although organizational process definitions do not necessitate these advanced features, they may be facilitated by them.

3.1. GOALS OF ORGANIZATIONAL PROCESS DEFINITION

Organizational process definitions need to provide all the advantages of sharing, while preserving the ability to deal with variations among projects.

Organizational process definitions are intended to convey both constraints and flexibility. You want to constrain the processes in such a way that you can pursue organizational goals while avoiding disasters, yet you also want to preserve—and pass through to your program and project managers—the flexibility they need to select *how* to efficiently and effectively engineer end-products. This allows you to vary a process to different circumstances even though the final product is essentially the same.

Consider the housing industry. One process for building a house involves hiring an architect, designing a totally custom house, and having it built to your exact preferences. A different approach is to go to a developer who might have a hundred or so predesigned types or styles of houses—some intended for use in cities, some designed for use on mountains, others for coastal communities. As a buyer, the process allows you to select from some “reusable designs” and tailor the design to suit your preferences. Tailoring options may be extensive, such as whether you want the optional garage, porch, or three-story model. The result of each of these processes is the same: you have a house that you expect to suit your needs. Similarly, you might also visit a builder who is constructing a townhouse community. In that scenario, you may have the option to select from six or seven basic models. For each model, your tailoring options could include whether or not you want a fireplace, upgraded carpet or appliances, a bay window for the living room, etc. Here too, you end up with the same product: a house to live in. But the process that produces and markets these houses is markedly different in each of the examples.

Independent of these three process variations for building and marketing houses, there are certain invariants that hold true for each of the processes. For example, there are electrical codes, building codes, standards for plumbing, structural stresses or load maximums, federal and state zoning and tax requirements, etc. These are, effectively, “organizational” process constraints that must be adhered to independent of the type of process being used.

As can be seen from this example, process definitions can—and should—occur at different levels of abstraction. The number of abstraction levels appropriate for you is a function of the size of your

organization and its type. However, as shown in Figure 3-1, one way to analyze levels is to think about process definitions at five levels of abstraction. These levels are:

1. Organizational process definitions
2. Business area process definitions
3. Program process definitions
4. Project process definitions
5. Project plans

Note that your options span the entire spectrum from the highest or least detailed level (organizational definitions) to the lowest or most detailed level (a workable plan that is used to enact and track the process). The remainder of this section elaborates upon issues related to developing process definitions at different abstraction levels. This section concludes with a brief discussion on how to develop organizational process definitions via the creation of “pockets of excellence.”

3.2. TOP-DOWN SOFTWARE PROCESS DEFINITION

The software industry is beginning to recognize the importance of standard and defined development processes to producing consistently high-quality software within budget and on schedule. However, unique project characteristics may impact a project's ability to perform to a previously defined development process: that is, different process drivers generally result in software development processes that may vary slightly to significantly from project to project.

Many process groups today are faced with the difficult task of defining the product development process for a large, complex, and highly variable organization, as well as for the multiple and diverse projects within that organization. Finding the right set of solutions requires structure and discipline, including careful analysis of the problem, designing an architectural framework within which the problem can be decomposed (a top-down perspective), and decomposing the problem into smaller, more manageable pieces.

Similarly, imagine trying to build a large, complex software system without first designing a software architecture: interface problems are prevalent, requirements are implemented inconsistently or not at all, and design constraints are violated. Like software engineering, one approach to process engineering is to represent complex interactions among process steps through a high-level process architecture that can be subsequently refined and elaborated. That is, process engineering can follow an approach which elaborates the process through different forms of representation and level of detail until an enactable, or performable, level is reached with appropriate reuse and tailoring at each level. (Discussion of a “bottom-up” or object-oriented approach is deferred until Section 3.5.)

Many software organizations have expended a significant amount of time and resources to produce voluminous process descriptions. Often, these descriptions do not correspond to the processes actually performed during software development or maintenance. Some key problems may cause the disconnect between how a project actually performs to produce its required product and the organization's standard process model, including:

- Organizational process models developed at too high a level of detail
- Organizational process models defined at too low a level of detail
- Organizational process models that mandate inappropriate life-cycle model(s)

Process Engineering With the Evolutionary Spiral Process Model (Software Productivity Consortium 1994) states that to a large extent, these key problems are symptoms of the difficulties currently encountered when attempting to tailor an organizational process definition for a specific project. To resolve tailoring problems, it may help to consider the process model in terms of reusable assets, including:

- Organizational Process Definition
- Business Area Process Definition
- Program Process Definition(s)
- Project Process Definitions
- Project Plans
- Process Asset Library
- Process Measures Database
- Policies
- Standards
- Program Assets
- Procedures and Methodologies
- Resource Mappings

You can gain additional insight by understanding how to develop and elaborate process assets based on different levels of drivers, such as organizational, business area, program, and project. Section 3.3 (and Figure 3-1) shows you a process through the different levels of elaboration, producing the supporting process assets.

3.3 PROCESS ENGINEERING

Process Engineering With the Evolutionary Spiral Process Model discusses the process engineering process in terms similar to those shown in Figure 3-1. The material in this guidebook slightly extends the approach to provide information on process definition abstraction at the program level.

3.3.1 PROCESS ENGINEERING PROCESS

Figure 3-1 shows activities to produce and continually evolve the appropriate process assets to the appropriate level of detail. In addition, it specifically surfaces the impact of process drivers that place requirements on each level of abstraction or elaboration.

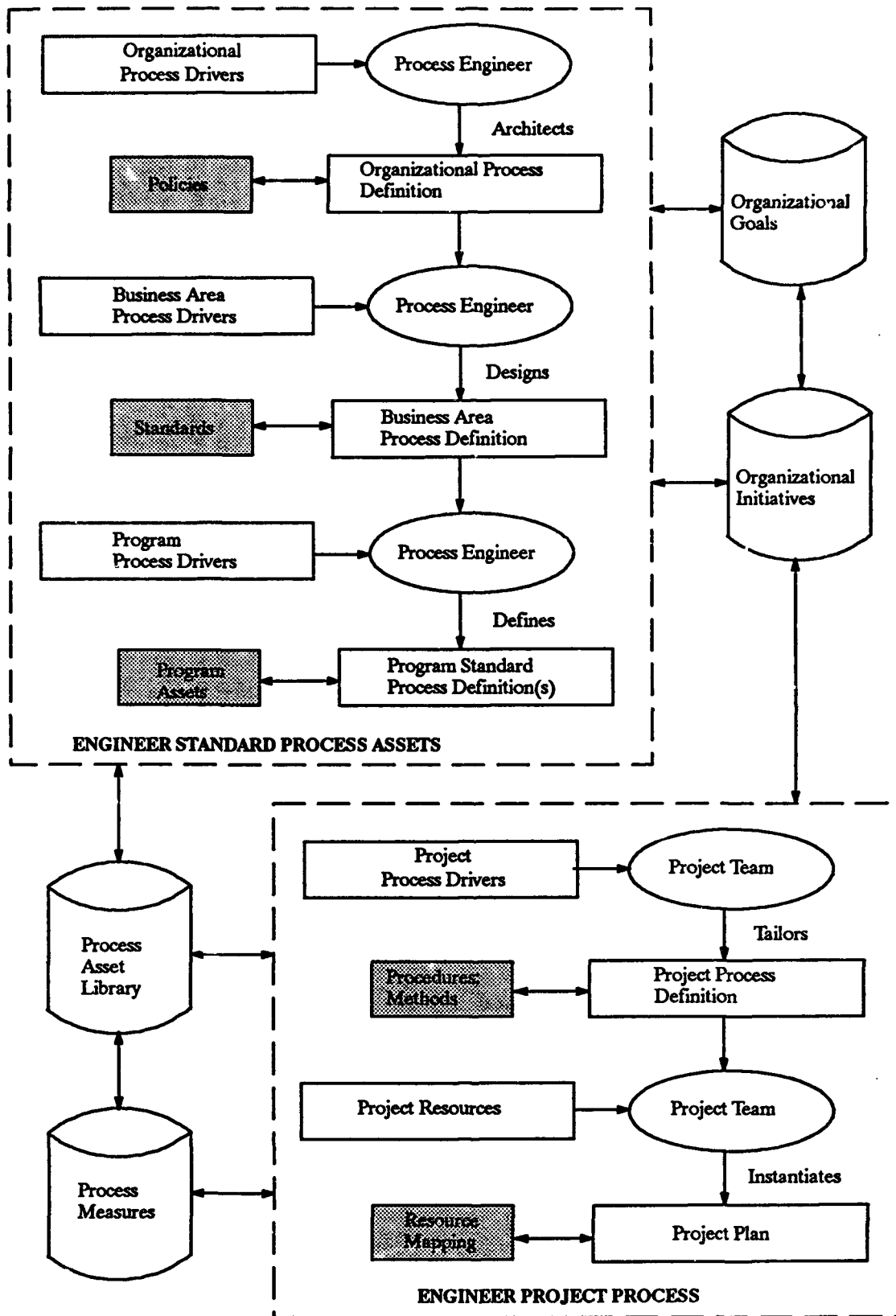


Figure 3-1. Abstraction Levels for Process Definitions

At the organizational level, top-down process engineering can start by identifying existing process assets, including activity specifications, method descriptions, standards, and artifact definitions, and using them as the basis for creating an organizational process definition. Parts of the process definition may be unique to a given business area. Where no appropriate material exists, you may need to create it from scratch.

An organization may be divided into two or more business areas. These are coherent markets characterized by customers possessing similar needs. In this case, you may find it useful to further detail one or more standard process definitions based on the unique drivers of the appropriate business areas.

Additionally, within a business area you may have two or more major programs occurring. Typically, each program will have requirements in common with all the other programs (and hence, can be considered business area drivers) and each program will have requirements that are unique unto the program. These are the program process drivers.

Project process definition addresses product-specific characteristics, such as the product risks and life cycle, and is based on the process definition developed for a specific program, business area, or for the organization (if the concepts of business areas and/or programs do not apply). The information identified in the program process definition is fully elaborated in the project process definition, which is turned into an enactable process plan by mapping people and other resources available for the project, establishing budgets and schedules, etc.

Sections 3.3.2 and 3.3.3 discuss each of these top-down process engineering life-cycle activities in terms of the key process assets that result from each and the process drivers that place requirements on the development of assets.

3.3.2 ENGINEER STANDARD PROCESS ASSETS

As shown in the top box of Figure 3-1, higher abstraction levels of process definition have the goal of producing standard process assets. This includes creating organizational definitions, business area definitions, and program process definitions.

3.3.2.1 Organizational Process Definition

This activity involves defining organizational context and organizational process policies. You analyze and document how organizational procedures are enacted currently and resolve conflicting interpretations of these procedures to obtain a consistent view of organizational policy. To synthesize this information at an organizational level of elaboration, the process is typically represented by a set of organizational policies, with appropriate cross-references to show policy relationships.

Organizational process definitions vary from one organization to the next, based on unique process drivers. Process drivers that may place requirements on the identification and definition of process policy at an organizational level include:

- Investments (such as hardware, software environments, training, etc.)
- International standards

- Infrastructure and culture
- Government rules and regulations (such as Federal Acquisition Regulations)
- Market trends
- Performance goals

If an organization is not further divided into business areas, then you should develop an integrated set of program process definitions that includes the information discussed in Section 3.3.2.3.

3.3.2.2 Define Business Area Process Definition(s)

This phase involves decomposing organizational policy and augmenting that process information with additional detail. Typically, business areas are subject to several internal and external standards, especially product standards. At this level, your process definition will need to communicate, at a minimum, the organizational policies and standards that drive one or more of the programs conducted within the business area.

To accurately develop business area process definitions, you must first determine what constitutes an organization, business area, etc. Instead of an entire company, an organization can also be a division or a functional area within the division. One company may feel that it has several divisions, each with sufficiently different missions, customers, products, and cultures to warrant creating several business area-specific process definitions. Another company may determine that one process definition is sufficient because only one type of business is pursued.

However, if the business areas have specific operational environments that are significantly different, then multiple business area process definitions can be developed based on business area unique process drivers, such as:

- Technology and business strategies
- Product lines
- Methods particular to the business area
- Risks specific to the business area
- Industry standards for the business area
- Existing product assets for the business area, such as development environments and reusable software

You can define specific business areas at a lower level of granularity than at the organizational level because of visibility into product lines of various programs within the business area. This visibility allows you to determine, select, or identify applicable process and product standards.

3.3.2.3 Define Program Process Definition(s)

At the program level, you can identify the type of products produced within the program, and hence, the life-cycle models for key products in the product line. You then begin to elaborate the life-cycle

models with partially ordered steps or activities that will help guide projects in achieving each of the key product's life-cycle states.

The process definition at the program level will facilitate construction, evolution, and product reuse by identifying and defining standard process steps at a summary level. The goal is to identify the partially ordered key process activities, which compose the overall process, and elaborate the primary relationships between them. The resulting definition is still at a high level of abstraction and, typically, has the following characteristics:

- Partially ordered sequence of process activities
- Key process activities that you define in terms of their functions, primary inputs, and major outputs
- Naming conventions and standards
- Standard process templates or formats
- Interface specifications
- Composition, selection, and tailoring rules

Again, at the program level, the process definition typically does not decompose the key process activities in any great detail. You generally represent it graphically and support it by text or templates that define the high-level scope, objectives, and function of each key activity, and the policies and standards that guide, cause measurement of, or constrain those activities.

3.3.3 ENGINEER PROJECT PROCESS

The result of the above types of process definition (Section 3.3.2) is a repository—typically called a process asset library—of reusable process assets. These assets are used to facilitate the development of project process definitions which are tailored to the needs and circumstances of individual projects and can be instantiated and enacted.

3.3.3.1 Project Process Definition

This is a two-phase activity which yields a preliminary and final project process definition, respectively. In the first phase, you work with the process manager to select the process asset(s) that will be used on the project to guide the key product's development phases. This preliminary project process definition is a description of a project's process and includes a selected life-cycle model. It identifies the various options for process steps, the options for performing them, their various connections to each other, and alternative mappings to the product life-cycle phases. The preliminary project process definition is developed based on specific product drivers, such as:

- Product-specific requirements
- Product-specific development standards

- Product-specific methods
- Product-specific risks, such as performance or integration problems

This phase can be relatively uncomplicated if an organization is subdivided into business areas and programs, and the program has defined its product line and the life-cycle models that best support each key product in the product line. If this is the case, there may be several preliminary process definitions "on the rack" for a project to choose from thereby eliminating the need to custom build the preliminary project process definition.

During the second phase of this activity, you use the preliminary project process definition to derive a final project definition. During this phase, you and the project manager elaborate the preliminary project process definition by selecting from among the alternatives presented in the preliminary project process definition. This is your opportunity to tailor the final project process definition specifically to the project situation. Generally, the final project process definition is developed by: selecting and tailoring the process step specifications that will meet the objectives of each product life-cycle stage; identifying the work products that will be produced by each step; and choosing the methods that will be used by each step. The final project process definition is tailored based on project specific drivers, such as:

- Selected product life cycle
- Project objectives, such as producing zero-defect software
- Project constraints, such as customer requirements or environment
- Project risks, or potential problem areas, such as limited access to users

The project manager can define the project process definition either before it is performed by the project team or in parts, as long as each part is defined prior to performance. In either case, the result should be a (progressively more) final project process definition that represents the roadmap that the project team will follow to reach each life-cycle state.

3.3.3.2 Instantiate Process Plan

In this phase, the project manager instantiates the final project process definition by using it as a template for the project schedule. The manager also uses the process step specifications as templates for specific project tasks, assigns specific agents to the each task, and estimates the time and effort needed to complete the tasks. Specifically, the project manager instantiates the final project process definition by assigning schedule and resources to produce an enactable process for the project team to follow. The project manager makes these allocations depending on such drivers as staff availability and experience, contract milestones, contract budget, and risk mitigation.

The result of instantiating the project process definition is a project plan. The project plan should document, or provide a reference to, everything that is needed to enact the project process. The process plan specifies the resources necessary for enacting the process, the relationships of these resources to process steps, the products produced by these steps, and any constraints on enactment or resources. Resources include human process agents, computer resources, time, and budgets. Relationships refer to the estimation or assignment of resources to process steps to meet project objectives (Feiler and Humphrey 1992).

Again, the final project process definition does not need to be instantiated completely at one time; often it is better to define and instantiate it in increments based on lessons learned and an increasing knowledge about project and product process drivers and awareness of changes in the circumstances surrounding the project as the project evolves.

3.3.3.3 Enact Project Process

To put the process into practice, the project manager performs resource leveling in order to validate that the proper resources have been allocated to each task, and initiates, monitors and controls the execution of each task according to the project schedule. An enactable process consists of a process definition, required process inputs, assigned enactment agents and resources, an initial enactment state, an initiation agent, and continuation and termination capabilities. An enacting process may be in a suspended state if an assigned process agent is not available or other process constraints are not satisfied (Feiler and Humphrey 1992). The project team enacts the project process by following the plan and performing their assigned tasks.

3.4 PROCESS ARCHITECTURE

Process architectures can be developed to support any level of process definition abstraction. You can have process architectures at the organizational level, business area level, program level, and even at the project level. As a rule, the higher the level of abstraction, the more sparse the architecture. Section 3.4 outlines what the purposes of a software process architecture are, what composes one, and what are many of the issues involved in creating one (Redwine 1991).

3.4.1 PURPOSES

While an architecture might appear to be just describing the structure and relationships among the components of a system and the system's interfaces with its operational environment, architectures actually fulfill a number of purposes. For software process architectures, these purposes include:

- Describing the significant components, structure, internal and external relationships and interfaces
- Defining graceful evolution paths and required reuse variations
- Guiding component selection, adaptation, composition, and binding
- Allowing smooth assembly of the components and connecting them with the surrounding environment
- Providing compatibility across multiple instances

An architecture must allow for different degrees of elaboration, functionality, and performance. An architecture should also, where possible, facilitate the provision and use of various components. Any architecture must also try to address particular qualities or properties important for its system; for example, a fault-tolerant structure might help address a requirement for high reliability.

A software process architecture is reusable across some subdomain of software projects. A reuse architecture aims at capturing the invariant plus providing for anticipated changes and variations. You

must start by deciding what can be treated as stable and what variety must be accommodated. Decisions having strong impact on an architecture include the scope of variety that must be accommodated in methods, role definitions, and standards. While many organizations may simplify their concerns by severely limiting the variety in these areas, you may have to treat them as having significant variety.

3.4.2 WHAT IS NEEDED IN AN ARCHITECTURE

A software project involves performing tasks that create and change artifacts using resources and technology in an organizational, professional, and market context. To be (re)usable and practical, the architecture must be reasonably integrated and contain guidance about how to tailor and instantiate in different circumstances. Thus, a software process architecture can accommodate the following kinds of components, each with appropriate levels of granularity:

- For tasks: Activities—input, output, entry and exit criteria, measurements, and control
- For artifacts: Data—objects attributes, relationships, structures, and constraints
- For fit and customization: Tailoring and instantiation guidance—selection, adaptation, composition, and binding
- For integration: Policy constraints and standards aimed at integration
- Possibly three other kinds:
 - Constraints on activity contents
 - Additional constraints on activity composition and timing
 - Organizational entities and constraints

Looking at a number of existing descriptions of “life-cycle methodologies,” or other descriptions that have been found useful, the number of activity types is generally between 50 and 200. Activities can be black boxes, possibly arranged in two to four levels of decomposition, to be filled by methods performed by tools and persons.

The level of detail in the architectural data schema or artifacts will vary depending on the invariance of the details and on whether the architectural schema explicitly details alternatives to a degree appropriate for the intended level of abstraction. Data schema are engineering-oriented; documents are only particular presentations. Constraints over the data can be either active or passive and must allow for inconsistency tolerance.

Instantiation guidance is needed to allow instantiations to be readily performed across a wide range of projects. Variations in projects that need to be accommodated at instantiation time include variations in not only the levels of understanding of requirements and architecture and availability of reusable assets (Boehm and Belz 1989) but also in other factors, such as performance, degree of distribution, standards, etc.

While policy constraints and standards can address a number of issues (particularly for organizational- and business-area level process architectures), integration issues must be dealt with

at least by the program level (that is, you may have already dealt with some integration issues at the business-area level; but if not, integration should definitely be addressed at the program level). This includes terminology, data integration, control integration, and method and tool interoperability. Note that some integration issues, such as terminology, may well be addressable above the program level. That is, your process architecture may show the role and use of terminology across an entire business area.

The three other possible kinds of constraints that may be included in a process architecture reflect either more convenient ways to describe items or ways to describe laws or policies. For example, it is convenient to say that inside each activity black box shall be tasks aimed at progress as well as verification (and possibly validation), measurement, and local management.

3.4.3 ISSUES

An architecture will be influenced by goals, object-oriented class and other structuring, organizational and human aspects, and automation. The kinds of goals that may impact an architecture include continuous improvement, exploiting application-specific attributes and knowledge, commonality across development and evolution, fault tolerance of the software process, early defect detection, risk reduction, measurement, reuse, and quality.

Activities, data, and methods can have class structures with inheritance. For example, an “evaluate” class might have subclasses of review, test, and analyze. “Explore” might have subclasses of prototype, model, collect data, and study. Other structuring issues include layering an architecture from general to detailed (e.g., policies at the top and procedures at the bottom); separation of concerns (e.g., functional, temporal, and organizational concerns); and units of reuse.

Organizational and human aspects include how projects are structured and managed, geographic decentralization, use of subcontractors, quality of personnel, and role structure. Process architectures must decide how much management, quality assurance and/or independent verification and validation, and measurement are separate, explicit, or invariant in the architecture. An architecture may address control and autonomy. In addition, business enterprise and system engineering processes and resulting constraints must be accommodated. When an architecture must accommodate significant variation in role definitions, then local role fragments or task-roles may be used to allow their composition into roles conforming to an organization’s definitions.

Human problem solving jumps within seconds across such boundaries as requirements, preliminary design, and detailed design. Process architectures must allow or even facilitate such opportunistic behavior—or they become only instructive myths or even barriers to accomplishment. While timing/synchronization constraints cannot be as severe as in the traditional waterfall method, constraints can usefully address visibility, atomicity, and configuration management.

Automation choices can come either before the architecture or at instantiation or enactment. One significant special case is the supplying of both automation and process architecture by the same source—a situation with significant potential for simplification or synergism.

Finally, one of a software process architecture’s most important impacts is the requirements it places on methods used to accomplish activities. Beyond fitting boundaries, methods may need to provide data details, progress and continuous-improvement measurements, and information for project planning and estimating. This is in addition to local management methods, appropriate verification

and validation methods, and conformance to integration standards. Finally, methods need to treat the many instantiation variables mentioned, such as the degrees of software systems' decentralization and dependability criticality. A method may be suitable or adaptable across a subrange of an instantiation variables.

At least initially, it is useful to have your architecture be silent in a number of more advanced areas. If done judiciously, this can simplify your task and allow you to speed it into use and gain experience.

3.5 THE POCKETS OF EXCELLENCE APPROACH

So far, this section has described process definition that can be approached in a top-down manner. There are, of course, countless other approaches. A substantially different approach is to build "pockets of excellence" within your organization by defining your process from the bottom up. Once you have "backed into" your organizational process definition, you can then reverse course to improve and optimize your process definitions in a top-down manner. Then, after reaching and improving project-level definitions, reverse course again and start working your way back up through the abstraction layers. Highlights of the bottom-up approach are presented in Sections 3.5.1 through 3.5.10.

3.5.1 IDENTIFY HIGH-VALUE, LOW-RISK PROCESS KERNEL

Typically, as a result of either a formal or informal process assessment, you will have identified several areas of your process that are candidates for improvement. This provides you an opportunity to create a pocket of excellence within those projects that were assessed. An important decision you need to make is which process problem are you going to address first. For example, do you want to improve the configuration management process, the quality assurance process, project tracking, project management, or peer reviews? Each of these process areas, when defined, yields a "process kernel" that becomes part of your process asset library.

When you are selecting which process kernel(s) to define, keep the following advice in mind:

- Prefer smaller process areas to larger
- Prefer simpler process areas to those that are more complex
- Prefer a process that is almost adequate, as opposed to one that is totally inadequate
- Prefer a scope of effort that is shorter as opposed to longer
- Prefer a process area that the projects consider unthreatening

To summarize, when you start defining your process, first select a process kernel that allows you to show a quality process definition in a short amount of time which has high likelihood of being accepted and used on the projects. As you gain in experience and ability, you can extend your effort to define more difficult or challenging process areas (as indicated in Section 3.5.7).

3.5.2 IDENTIFY PROJECTS FOR POCKET OF EXCELLENCE

Once you have identified the process kernel you want to define, you need to identify the initial audience who you expect to use the intended process definition. Generally, the audience will consist

of one or more projects which were included in the original process evaluation or assessment. It can certainly be the case that another project, not included in the original assessment, may be a good candidate for the newly defined process. However, to whatever degree that external project feels it is "different" from those assessed, you likely find it correspondingly more difficult to convince them the newly defined process is applicable to their circumstances. Regardless of how similar you think projects are, the managers of each of those projects will, almost invariably, consider their project unique.

Try to find projects that truly are interested in incorporating the newly defined process. As incentive, remind them that their advice will be unobtrusively solicited (as described in Sections 3.5.3 through 3.5.5) throughout the process definition effort. Keep in mind that you are not necessarily defining a new process for them to use—the goal of this particular phase may well be to simply define the process as it is currently being performed.

3.5.3 INTERACTIVELY DEFINE COMMON PROCESS KERNEL FOR TARGET PROJECTS

It is almost imperative that you involve your end-users in the work of defining their process. Involvement includes, at a minimum, conducting interviews, even if they are quite brief, on how they do their work and on how they think it should be done. Have the end-users (at their discretion—but try and encourage participation) review and comment upon various draft releases of the process definition. Later, continue to involve your users by maintaining an "open door" policy for ongoing feedback after a version of your process definition has been baselined and officially released for use. Also, visit them to determine their perceptions about and use of the process kernel definition.

3.5.4 PILOT INSTANTIATION OF PROCESS KERNEL ON SELECTED PROJECT

As you develop the definition of the selected process kernel, you will perceive different degrees of interest and support from among the projects involved in becoming a pocket of excellence. It is generally the case (but certainly not always) that the project that offers you the most support or shows the most interest is also an ideal candidate for piloting the newly defined process. The purpose of the pilot is to acquire that last degree of insight based on feedback from "real-world" application.

To whatever degree possible, try to pilot the newly defined process kernel on a project that:

- Is being managed by someone who is interested in piloting the newly defined process
- Is staffed by people who are interested in piloting the newly defined process
- Is not under unusual time constraints
- Is not under unusual budget constraints
- Is not already involved in significant change
- If a complex process kernel, has successfully piloted prior newly defined processes

In practice, however, the overriding factor that drives success is whether you can find a project where the people want to participate in piloting the newly defined process.

3.5.5 BUILD VALID PROCESS DEFINITION

Piloting the newly defined process will invariably yield insights into the applicability, usability, and intrinsic value of your process definition. Use the pilot as a means for continuing to improve the overall quality and value of the process definition. Do not make substantial changes to the process definition without validating the impact and value of those efforts on your pilot project. If for some reason, the project completes before you are done with the process definition, or if the project must, for whatever reason, disengage itself from the pilot effort, then be sure to find another pilot project. It is extremely important that you validate your process definition before releasing it to your general audience.

3.5.6 TRANSFER VALIDATED PROCESS KERNEL TO OTHER PROJECTS IN POCKET OF EXCELLENCE

Once you have a defined process kernel validated, your next step is to export that definition for use on other projects targeted for the pocket of excellence. Do not just ship out guidebooks. For each project you have targeted, attempt to arrange with the project manager for a “roll-out” meeting of the new definition. This can be as short as 10 minutes if you defined a process the project was already trying to follow. For duration, we highly recommend the roll-out meeting not exceed 30 minutes. The roll-out is **not** intended to train people on the process. It is only intended to make the project aware of:

- Who you are
- What you are doing
- Why you are doing it
- How it is intended to help them
- How they are expected to use what you are providing them
- How they can reach you if they have questions, comments, concerns, or recommendations

After the roll-out meeting, you will need to aggressively follow up and support the project—and possibly help the project manager arrange for training. Ideally, you will find that the project accepts, absorbs, and appreciates having the process definition available. However, in those instances where they find using the process definition to be awkward, excessively time consuming, or confusing, you will certainly want to be aware of that, and take steps to address these issues at the earliest opportunity.

3.5.7 SELECT NEXT PROCESS KERNEL

Generally, unless you have a high degree of expertise, we discourage you from attempting to define several process areas simultaneously. This is especially true if one or more of the process areas are complex or involve areas of possible contention regarding “best” approach. You should first secure two or three solid—and sequential—victories.

When you are reasonably certain that your first process definition has high quality and will be of value to the projects, you can begin work on the next process area. The process proceeds in much the same way as described above, with the exception that in time you will have the expertise to confidently address two or more process areas as part of a single process definition effort.

3.5.8 SELECT NEXT POCKET OF EXCELLENCE

Once several process areas have been defined and transferred into the projects within a particular pocket of excellence, you will want to look for other areas within your program or business area that can benefit from the existing set of process kernels. For each new pocket of excellence, you should select a project to pilot the newly defined process kernels. You need to verify that the new pocket of excellence is indeed similar enough to the original pocket of excellence. If the similarity is sufficient, projects in the new pocket of excellence will very likely perceive the process definition(s) to be applicable to their situation, valuable, and usable. If the definitions are not accepted by the projects, it may be because the new projects have process problems in different areas or the new projects are too different. These issues are addressed in Sections 3.5.9 and 3.5.10, respectively.

3.5.9 DISTILL PROCESS KERNEL DEFINITION(S) FOR MULTIPLE POCKETS OF EXCELLENCE

At a certain point, as you continue to export defined processes to more projects, you will find that there is a need to address new problems that did not surface during your original process assessment. Certainly, this may indicate that it is a good time for another assessment. However, it is also true that you do not need to wait for an assessment before you define a process. All that is necessary is for the organization to acknowledge that process problems can at least partially be addressed by developing or improving upon the definition of one or more process areas and then authorizing that work.

This work proceeds as described in Section 3.5 (selecting one or more process areas, interactively defining them with the involvement of a subset of your intended audience, piloting/validating, etc.). This work leads to continuing extension and expansion of your process asset library.

3.5.10 ABSTRACT UP TO ORGANIZATION-WIDE PROCESS POLICIES

Projects will also resist a given process definition if they consider the definition to be inapplicable given local project circumstances and characteristics. In other words, the defined process is not sufficiently tailorable to that project's needs.

This becomes the catalyst for you to evolve your process definitions to progressively higher (less detailed, more flexible) levels of abstraction. Note that as a general rule, you distill higher levels of abstraction by **throwing details away**, not by adding more details. The level of detail and the number of necessary constraints are less at the program level than they are at the project level. Similarly, the level of detail and the number of necessary constraints you need at the business area level are less than what you need at the program level.

As implied by Figure 3-1, you can start at the bottom, or most detailed level, and define your organizational process as follows:

- Take your project plan. Throw away the names of people, dates of work, and similar enactment information, and you have a good foundation for your project-level process definition.
- Take your project definition. Throw away information that is specific or unique to that particular project, and you have a good foundation for program-level process definition.
- Remove program-specific information and life-cycle models and you will likely be left with two major types of information: standards with which the process must comply and organizational policy. Jointly, these serve as a good foundation for business area process definition.

- If you eliminate standards that are relevant only within, but not between, business areas, then the remaining information—typically policy intensive—can be an excellent foundation for an organization-level defined process.

As stated at the beginning of this section, the five levels of abstraction used in this material may not be applicable to your organization. You may, in fact, be quite comfortable defining your processes at two levels: organization and project. Nevertheless, the key point remains the same: at every level of abstraction, strive to define only what has to be defined. Keep your process definitions as Spartan as possible, and strive to preserve maximum flexibility at every level of abstraction. This permits your project managers and engineers the greatest opportunity to leverage their experience, expertise, and talent for addressing unexpected problems and for capitalizing on unforeseen or unusual opportunities.

4. ADVANCED TEMPLATES

Prior sections have examined a variety of global, high-level issues related to establishing, managing, or participating in a program of process definition. Critical aspects of process definition are collecting and managing your process information. As discussed in Volume 1, Section 3, process definition is truly an information management problem—not a word processing problem. Consequently, it is recommended that you use an information management environment to support your process definition effort and then “export” process information from your database for use by other automated tools (e.g., word processors, presentation packages, etc.).

Section 3 introduced the concept of the MPDM templates. Section 4 elaborated on details on how to use the process templates, and Section 5 presented an extensive example on how process information can be algorithmically extracted and exported to create “push-button” guidebooks, training material, etc. This section considerably expands upon the information model presented in Volume 1. However, general principles on template usage remain as presented in Volume 2.

4.1 ADVANCED TEMPLATES OVERVIEW

The three templates discussed in Volume 1 capture only fundamental process information. As you can see in Figure 4-1, the full set of MPDM process templates consists of 37 templates at 4 levels of abstraction. Specifically, MPDM templates consist of:

- One Foundation template
- Five Meta-Class templates
- Two Class templates
- Thirty-one Subclass templates

Before discussing details of these templates, there are two important concepts that need to be reviewed. First, the foundation, meta-class, class, and subclass templates represent an inheritance structure that shows you the scope of use for any given field. If a field is shown on the foundation template, that indicates it exists on all subclass or “final” templates. Meta-class templates add additional fields that pass to all the class (and subclass) templates within that meta class. Class templates pass their fields through to the subclass level. In all cases, you eventually arrive at subclass or final templates—these are the templates you actually use in gathering process information.

Second, the fields distributed among these templates allow you to collect and manage information at several levels or “tiers” of usage. As you may recall from Volume 1, MPDM recognizes four tiers of process definition. Each tier has different goals. Note that tiers of usage are independent from levels of abstraction discussed in Section 3. Typically, however, your intended usage will be less formal at

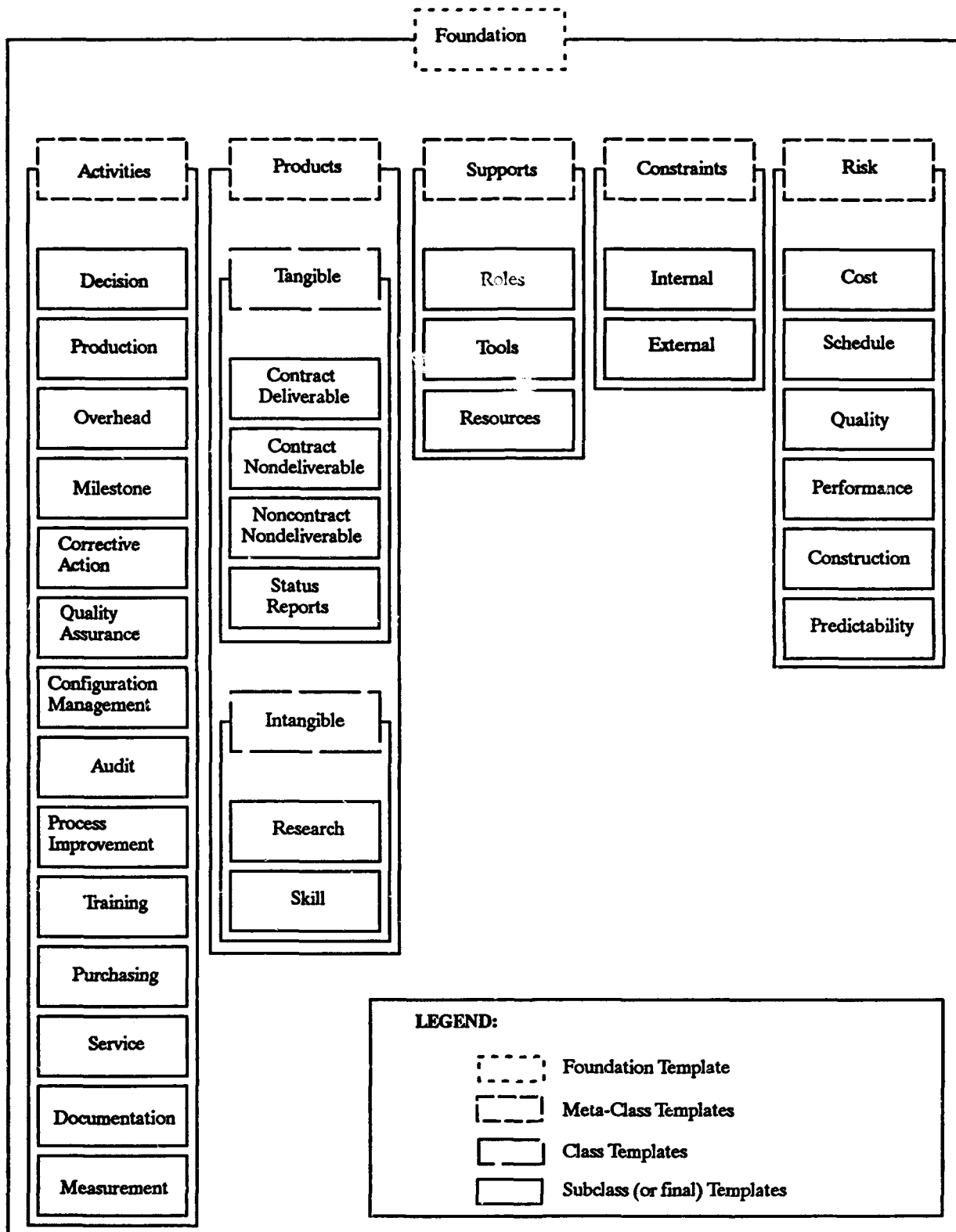


Figure 4-1. Template Inheritance Model

higher levels of abstraction and, if you have a need for increased formality, it will first occur at the program or project abstraction levels. For example, you will likely need an enactable model of one or more project processes before you need an enactable model of an entire business area.

To briefly review the concept of tiers from Volume 1, higher tiers of process definition require increased formality, and lower tiers require less formality. The four tiers are:

- **Tier 1: General.** The goal of Tier 1 usage is to create and maintain process models that facilitate the production of process-related end-products that must be usable by large, general audiences. Examples include guidebooks, training material, and process-related sections of contract proposals.
- **Tier 2: Procedural.** The goal of Tier 2 usage is similar to Tier 1, except that more detail is desired. Typically, this reflects the evolution of guidebooks that convey progressively more detail. Operations manuals, for example, are process end-products that typically contain high degrees of "how-to" details.
- **Tier 3: Formal.** The goal of Tier 3 is to provide end-products that facilitate static analysis of the process. The purpose of such analysis can be to discover desirable or undesirable process characteristics, identify areas of unusually high process risk, etc.
- **Tier 4: Enactable.** At Tier 4, the goal is insights into the behavioral characteristics of a process, whether virtual or real. Currently, this is primarily achieved through automated process simulators and enactment environments, respectively.

In the interest of brevity, fields that were discussed in Volume 1 of this guidebook are not repeated here. Also, in many cases, a group of fields presented as Tier 1 will have additional fields that are at Tier 2, 3, or 4. Again, only the non-Tier 1 fields are discussed.

Many of the field names in Section 4.2 are either self-evident or used in a manner consistent with a prior explanation. For example, Elaboration Text fields exist as part of many different groups of fields. In all cases, the use of this field is the same: it allows a place for you to add additional details about the group of data under discussion and how it relates to that template. To avoid duplication of material, once a field or group of fields has been explained, they will not be reexplained on subsequent templates unless there is something that makes their usage or purpose different.

There are some template classes, especially under the activity meta-class, that do not add any unique fields. These templates are composed entirely of fields inherited from the foundation template and the activity meta-class template. There are two circumstances in which you should use these templates. First, use them whenever it is important to show that a particular type (class) of activity is occurring in your process. For instance, the Corrective Action template does not add any unique fields. However, certain quality initiatives consider the ability to perform corrective action as a key indicator of process quality. When you want to highlight the fact that your process has explicit activities for taking corrective action, use this template.

Second, although some templates currently do not add additional fields, they provide a simple means to do so in the future. This improves your opportunity for organizing and extending the information you capture. If you anticipate adding additional fields, then having these templates already separated in your data schema can facilitate allocating, arranging, and managing that information.

Section 4.2 presents the advanced templates and fields involved in achieving the goals of Tiers 2, 3, and 4. Each field or group of fields indicates the Tier to which it applies.

4.2 ADVANCED TEMPLATES

The advanced usage templates are presented in the following order. The contents of the foundation template are discussed first. Each meta-class is then discussed; within each of these areas, the subordinate class (and subclass) templates are described. Meta-class templates are presented in the following order: activity, product, support, constraint, and risk.

4.2.1 FOUNDATION TEMPLATE

Virtually every field on the foundation template applies to Tier 1 usage. The few exceptions are shown below, and include additional information on reviews, access permissions, and project identifiers.

4.2.1.1 Review History (Group of Fields)

Review history fields are used to collect information about the occurrence and results of reviews. This group includes the following fields:

- Review ID: Tier 1 (multiple occurrence)
- Review Date: Tier 1
- Version Reviewed: Tier 1
- Review Composite Result: Tier 1
- Reviewed By: Tier 2 (multiple occurrence)
- Individual Review Result: Tier 2
- Individual Reviewer Comments: Tier 2

All but the last three fields are discussed in Volume 1. As you move into Tier 2 usage, you may find the following additional detail to be valuable:

Reviewed By is a multiple occurrence field that contains the names of everyone participating as a reviewer. Each name is treated as a unique identifier.

Individual Review Result indicates how each reviewer evaluated the product. For example, if five people reviewed a product, three of them might choose to recommend it be Passed Conditionally, while the other two might choose to recommend it be Failed Marginally. In such cases, either consensus opinion or policy would determine what the review composite result is. This field allows for distinguishing reviewer results prior to the consensus or composite result.

Individual Reviewer Comments provides a means for reviewers to provide a brief text comment back to the developers. This can be particularly useful if the reviewer has one or more solutions or recommendations that might be of value to the developers.

4.2.1.2 Access Permissions (Group of Fields)/Tier 2

This group of fields consists of:

- Owner
- Group
- World

Each of these fields contains one of the following three values:

- Read and Write
- Read Only
- Access Denied

This information can be used to control who has access to viewing and/or altering the information contained on the template. If your database or environment already provides for security, then you do not need these fields. Otherwise, you can begin to impose at least a nominal level of security (particularly to prevent accidental change) through the use of this group of fields. Note that if the Assigned To group of fields (discussed in Volume 1, Section 3) is empty, then, in principle, no one can access this template until your process database administrator provides a name and/or group to the Assigned To fields. This can be avoided by assigning completed templates to (for instance) the SEPG leader's name. This allows others in the SEPG group to retain access and also permits the SEPG leader to reassign the template or adjust permissions as necessary.

4.2.1.3 Project Invocation ID/Tier 4

To enact a process model typically involves "instantiating" a tailored subset of templates which define the process to be followed by a specific project. To distinguish this unique instantiation, each template will need to have a Project Invocation ID that identifies the project using this instantiation of the process.

4.2.2 ACTIVITY TEMPLATES META-CLASS

At the activity meta-class level, several new fields are added which apply across all classes of activities. There are fields targeted to support usage at each of the three higher tiers. The fields are organized by type of information, not by usage tiers. As you read, be sure to note the tier indications.

4.2.2.1 Objective/Tier 2

This is an open text field that is used to augment the information in the Description or Purpose group of fields (inherited from the foundation template). Typically, this is a more concise representation of that information and specifically focuses on the objective of the activity.

4.2.2.2 Abstraction Level

Activities occur at many levels of abstraction. The purpose of this field is to facilitate collecting sets of templates that contain, for instance, very detailed "how to" steps, very high-level policy statements,

etc. These can then be referenced by other templates, at different levels of abstraction, with the confidence that the desired level of detail is available. You should distinguish the following five levels of activity abstractions:

- **Process.** Process indicates an activity that does not have discreet start and stop times, but instead is ongoing. For example, "management" can be described as a process, whereas "review resumes" can be an activity.
- **Activity.** Activities have start and stop times, but typically lack "how-to" details.
- **Methods.** Methods are sequences of steps that describe, virtually in cook-book style, how to accomplish a particular task. Methods may involve one or more techniques.
- **Techniques.** Techniques are practices that are generically useful but which usually are combined with other techniques in order to build a "standalone" methodology.
- **Examples.** Examples indicate templates that provide examples of how to do work.

Obviously, there are no discrete transitions between activities and methods, or methods and techniques. Nevertheless, it is quite desirable to be able to model an activity, to show that the activity involves using one or more methods and techniques, and to show that an example or several are available.

4.2.2.3 Activity Criteria and Process (Group of Fields)

This group of fields is discussed in Volume 1, Section 3.

4.2.2.4 Formal Activity Criteria and Process (Group of Fields)

This set of fields is similar to the Activity Criteria and Process group of fields except for one key difference: *formality*. Whereas the Activity Criteria and Process group of fields contains text-based information intended to be easily read and understood by humans, the Formal Activity Criteria and Process group of fields contains information that can be easily read and understood by computers. Whether you use a process programming language, state transition language, first-order predicate logic, or some other alternative or hybrid, is a function of the capability of the enactment tool you are using. Use these fields to formally capture information that can be exported for use by an enactment tool. The four fields that compose this group are:

- Entry Criteria: Tier 4
- Internal Process: Tier 4
- Exit Criteria: Tier 4
- Invariants: Tier 4

4.2.2.5 Related Products (Group of Fields)

This group of fields is partially discussed in Volume 1, Section 3.

When relating products to activities, it can be useful to capture additional information about where that product comes from, where it is going, why you need it, etc. The following fields extend the Related Products group of fields to capture this information:

- **Source: Tier 2.** The Source field is used to simply distinguish where a product originates from. The suggested values for this field are:

- External
- Internal

Use **External** to indicate any product coming to an activity directly from “outside” the process. Use **Internal** to indicate a product that is generated as output from a prior activity in the process.

- **Destination: Tier 2.** The Destination field indicates where products go after this activity is done. The values for this field are:

- External
- Internal

Use these terms as described under Source field.

- **Purpose: Tier 2.** The Purpose field gives some idea of why the activity needs that product. Example values for this field are:

- Development
- Evaluation
- Reference

Development indicates the activity participates in developing the product. Evaluation indicates the activity is reviewing or auditing the product, but otherwise leaving it alone. Reference indicates that the product is being used as a reference or example, and hence, the product in some way influences, but is not changed by, the activity.

- **Product Level of Quality Needed: Tier 3.** Product Level of Quality Needed is used to indicate the degree of quality that must exist in a product before it is usable by this activity. For instance, early prototype activities can likely tolerate lower quality requirements. However, activities typically require higher quality requirements as they become more development-oriented. This field can be especially useful when modeling “spiral” processes. Suggested field values are:

- Very Low
- Low
- Medium

- High
- Very High

Clearly, the assumption is that these levels are qualitative, as opposed to quantitative, values.

- **Product Criticality: Tier 3.** Product Criticality is a field that ranges from Very Low to Very High (as described under Product Level of Quality Needed) indicating how critical this product is to the activity. During process simulation or enactment, this allows you to define, evaluate, and enforce complex heuristics for activity entry and exit criteria. Typically, these heuristics would, for example, evaluate the criticality of a product, in combination with its quality, to determine whether that product is ready to be used by the activity.

4.2.2.6 Related Supports (Group of Fields)

This group of fields is partially discussed in Volume 1, Section 3. However, there is additional information desired for Tiers 2, 3, and 4. Keep in mind that support can be either a role (a person or collection of people), a tool, or a resource. The additional fields are:

- **Minimum Required: Tier 2.** Minimum Required is a cardinality field that is used to indicate the minimum number of resources or roles required to support the activity.
- **Preferred Allocation: Tier 3.** Preferred Allocation is an ideal or preferred level of support, in terms of number of people, tools, or resources.
- **Maximum Allowed: Tier 3.** Any nonzero value in the Maximum Allowed field indicates the maximum number of resources or roles allowed.
- **Shared Access Needed: Tier 3.** Shared Access Needed is a value that ranges from 0 to 100% indicating an estimate of how much of the time the activity needs the support on a shared basis.
- **Exclusive Access Needed: Tier 3.** Exclusive Access Needed is also a value that ranges from 0 to 100%, but it indicates an estimate of how much of the time the activity needs the support on an exclusive basis.
- **Level of Support Efficiency Needed: Tier 3.** Level of Support Efficiency Needed is used to indicate the degree of efficiency that is expected from the role or resources when it is applied to this activity. As before, with product quality, you should use a system with five efficiency levels: Very Low, Low, Medium, High, and Very High.
- **Support Criticality: Tier 3.** Support Criticality is also a field that ranges from Very Low to Very High indicating how critical this support is to the activity. As with product criticality, this allows for complex heuristics about activity entry and exit criteria.

The Duration Information fields (see Section 4.2.2.11) assume that the Level of Support Efficiency Needed is provided. When efficiency levels are less than what is needed, your calculation (or simulation) of span or duration times should be proportionately increased. Likewise, higher levels of efficiency should result in shorter simulated durations. Use the Support Criticality field to adjust how much you allow the efficiency of a particular support

to affect the duration time of the activity. The efficiency of a support with very high criticality should have considerably more influence on an activity's duration than the efficiency of a support with very low criticality.

- **Current Access Type: Tier 4.** Current Access Type indicates the type of “lock” the activity currently has on the support. The two values are:
 - Shared
 - Exclusive

Exclusive conveys that the role, tool, or resource is not currently available for use by any other activity.

4.2.2.7 Risks (Group of Fields)

It can be very useful, even as early as Tier 2 usage, to begin to model the various risk factors that apply to activities. The fields that compose this group are:

- **Unique ID: Tier 2.** Unique ID is a unique identifier from a risk template.
- **Maximum Risk Tolerance: Tier 2.** Maximum Risk Tolerance is a number, typically less than 1, that indicates how much risk you are willing to tolerate before corrective action is taken.
- **Applies To: Tier 2.** Applies To indicates to which phase of the activity the risk applies. Field values are:
 - Entry
 - Internal Process
 - Exit
 - All
- **Risk Likelihood: Tier 3.** Risk Likelihood is a number from 0 to 1 indicating how likely you think this risk will manifest.
- **Risk Severity: Tier 3.** Risk Severity is a number from 0 to 1 indicating the consequences of this risk occurring. Generally values closer to 0 indicate low severity, and values approaching 1 indicate disaster.
- **Risk Frequency: Tier 3.** Risk Frequency reflects how often you expect this particular type of risk to recur. This value represents how often you think the risk is likely to occur on a “per cycle” basis. Twice per cycle is frequency 2.0. A likelihood of once every couple cycles indicates a frequency of 0.5. This refers to process risk, not product risk. Since processes have behavior in time, the risks associated with those processes also have time characteristics. When all else is equal, a risk that threatens us weekly is distinctly different from one that threatens annually.
- **Elaboration Text: Tier 2.** Elaboration Text is a free text field that allows you to describe characteristics of the risk and how they specifically apply to this activity.

- **Current Risk Likelihood: Tier 4.** Current Risk Likelihood is a fluctuating value that reflects your current best assessment of risk exposure as the project work occurs.
- **Current Risk Severity: Tier 4.** Current Risk Severity likewise reflects any changes to your impression of how severe a given risk is. Keep in mind that these are Tier 4 fields. They represent “actual” evaluations as opposed to those used during process definition or project planning.
- **Current Risk Frequency: Tier 4.** Current Risk Frequency shows the current evaluation, as a project progresses, of how frequently you think a risk exposure will occur.
- **Delta Below Maximum Tolerance: Tier 4.** Delta Below Maximum Tolerance is a calculated field. Its value is:
Maximum Risk Tolerance—(Risk Likelihood * Risk Severity * Frequency).
If this calculation results in a negative number, you have exceeded your risk tolerance.

4.2.2.8 Constraints (Group of Fields)

Constraints are anything that influences your process, but which are not better represented using one of the other template classes. For instance, you might want to capture the fact that an activity must be compliant to a particular standard. You may find it clarifies your model to represent standards as constraints. This group of fields consists of:

- **Unique ID: Tier 2 (multiple occurrence).** Unique ID is the unique identifier of a constraint template.
- **Applies To: Tier 2.** Applies To indicates which part of an activity is subject to the constraint. Options for this field include:
 - Entry
 - Internal Process
 - Exit
 - All
- **Elaboration Text: Tier 2.** Elaboration Text allows you to specifically describe, in free text, how the constraint applies to this activity.
- **Currently In Effect: Tier 4.** Currently In Effect, a Tier 4 field, indicates whether the constraint is currently active. Keep in mind that internal constraints are those for which you can receive a waiver. If the constraint has been waived, it is no longer in effect. Values for this field are:
 - Yes
 - No
 - Partially

4.2.2.9 Activity State Information (Group of Fields)

Product states were introduced as Tier 1 fields. However, it is also quite useful to describe activities in terms of states. This group of fields consists of:

- **State System Identifier: Tier 3 (multiple occurrence).** State System Identifier is a unique identifier for a set of states. Note that for any given activity, it may be subject to two or more systems of state transitions. For example, there may be one system of states that represents activity transitions from an engineering perspective. There may be another system of states that represents activity transitions from a management perspective. Still another state system could be used to define activity transitions from an auditor's perspective. Keep in mind that complexity in a process model invariably introduces risk of misunderstanding, mismodeling, and miscommunication. Therefore, you should only create multiple state systems in those instances where doing so simplifies, as opposed to complicates, your model.
- **Label: Tier 3 (multiple occurrence).** Label is the unique identifier for a particular activity within a cohesive set of states within one state system. You will need a unique label for each distinct state within that state system.
- **Description: Tier 3.** Description describes the state.
- **Application: Tier 3.** Application indicates whether this set of states should only be used at the current level of abstraction, or whether it is passed through inheritance to all the children (at all sublevels) under this template. The values for this field are:
 - Node Only
 - All Children
- **Augments: Tier 3.** Augments is a "Yes/No" field that is only relevant when a set of states has been inherited and when different state information is available at this level. A value of "No" in this field indicates the local state information completely replaces any inherited information. A value of "Yes" indicates that the local set of states extends or augments the inherited set.
- **Current Activity State(s): Tier 4.** Current Activity State(s) indicates, in an ongoing enactment of the process, what the current state is of this particular activity. Note that if you have defined multiple state systems, then this is a set of fields, one field for each state system.

4.2.2.10 Process Quality Attributes (Group of Fields)/Tier 3

Depending on your needs, values for the following fields can be either qualitative or quantitative. Keep in mind that these are process quality fields. Therefore, these fields are intended to capture information on process reliability, process efficiency, process safety, etc. This group of fields consists of:

- Reliability
- Efficiency

- Effectiveness
- Predictability
- Safety
- Error Density
- Fault Tolerance
- Probable Correctness
- Availability
- Maintainability
- Evolvability
- Security
- Survivability
- Utility

As with other qualitative judgements, we recommend that, at most, you use a five-level scale: Very Low, Low, Medium, High, and Very High. By rating your activities using some or all of these attributes, you allow for choosing between process options as a function of these attributes and their relative importance to a particular project.

For consistency, you only set these fields at final templates, and heuristically derive appropriate values for any contributing templates. Generally, span or duration times of child activities can be used to weight the emphasis given to a child's attributes.

4.2.2.11 Duration Information (Group of Fields)/Tier 4

This group of fields is used to capture the duration of a given activity. This information is essential at Tier 4 if you want to investigate simulated execution of your process. You will want to adapt this set of fields to match the conventions used by your simulation environment. However, example information needed by many tools includes average duration and how much this activity varies from the average. Certainly, standard deviations can be used. However, you can allow for more unbalanced variation by using lower and upper "high confidence" thresholds, as shown in the following list:

- Average Duration: Tier 3
- Typically More Than: Tier 4
- Typically Less Than: Tier 4

4.2.2.12 Planned and Actuals (Group of Fields)

This group of fields provides you a means to capture both what you plan for an activity and what actually occurs. This group consists of:

- Planned Earliest Start: Tier 4
- Planned Latest Start: Tier 4
- Planned Earliest Finish: Tier 4
- Planned Latest Finish: Tier 4
- Actual Start: Tier 4
- Percent Complete: Tier 4
- Actual Finish: Tier 4
- Critical Path: Tier 4

The earliest and latest start and finish fields support project planning and enactment. Of course, actuals can vary from planned, so there are fields to capture actual start and stop dates. Percent Complete is a value that ranges from 0 to 100 and can be used to facilitate analysis of project progress. Note that at the final templates, this information needs to be provided by someone. On any contributing template, it can be calculated as a function of weighted percent completion of the children. Critical Path can be derived algorithmically and contains either a Yes or No value.

4.2.3 ACTIVITY/DECISION TEMPLATES/Tier 2

This class of templates is used to represent the key decision points of your process. At a minimum, we recommend you collect additional information which characterizes the source(s) of information used to support the decision process.

4.2.3.1 Source of Information, Planned (Group of Fields)

The following fields all contribute to evaluating the source of information upon which the decision is based.

- Source Unique ID: Tier 2 (multiple occurrence)
- Expected Declared Accuracy of Information: Tier 3
- Expected Reliability of Source: Tier 3
- Expected Timeliness of Information: Tier 3
- Elaboration Text: Tier 2

Expected Declared Accuracy of Information represents what you expect to be the source's evaluation of the accuracy of the information. For example, you might typically expect a source of information to consider its information to be 95% accurate.

Expected Reliability of Source represents the confidence level you typically associate with that source of information.

Expected Timeliness of Information is a five-tier scale:

1. Very untimely
2. Somewhat untimely
3. Questionable timeliness
4. Somewhat timely
5. Very timely

As a rule, the more untimely information is, the less useful it is.

4.2.3.2 Source of Information, Actual (Group of Fields)

- Source Unique ID: Tier 4 (multiple occurrence)
- Declared Accuracy of Information: Tier 4
- Reliability of Source: Tier 4
- Timeliness of information: Tier 4

These fields are direct counterparts to those described in Section 4.2.3.1. The only difference is that, at Tier 4, you want to compare real-world values with those you had planned.

4.2.3.3 Minimum Threshold for Decision (Group of Fields)

There are two fields in this group:

- Text Description: Tier 2
- Numeric Minimum: Tier 4

Generally, you at least want to be able to describe, using the Text Description field, the minimum standards on accuracy, reliability and timeliness of information for the decision process to occur, particularly for Tier 2 usage. For either simulation or enactment, you will also need to use the Numeric Minimum field.

4.2.4 ACTIVITY/PRODUCTION TEMPLATES/TIER 2

This class of templates is used to represent activities that are directly billable. Consequently, this template adds a field to capture charge code information.

4.2.4.1 Project Charge Code/Tier 4

Although you might find this field valuable at lower Tiers of usage, for project enactment, you will want to capture the charge codes to which activities are billed.

4.2.5 ACTIVITY/MILESTONE TEMPLATES/TIER 2

Milestone templates are used to highlight key events within your process model. It is recommended that you use the convention that milestones are events and, hence, they do not take any time to occur. This can be easily represented using only the exit criteria fields on this template.

4.2.6 ACTIVITY/OVERHEAD TEMPLATES/TIER 3

Overhead templates are the counterpart of production templates. These are activities where you do not have project-direct charge codes.

4.2.7 ACTIVITY/CORRECTIVE ACTION TEMPLATES/TIER 3

The corrective action class of templates is used to highlight how you recover from problems. The use of these templates is exactly like the use of Exception Handlers in Ada. For simulation purposes, if a problem (such as noncompliant product) manifests within an activity, that activity should have a mechanism for addressing the problem. If not, the problem "propagates" to the next higher level. Ideally, at some point your process has a "corrective action" mechanism for handling the problem in a planned and predictable manner.

4.2.8 ACTIVITY/QUALITY ASSURANCE TEMPLATES/TIER 3

Quality assurance means a variety of things to different people. However you use this template, be sure to define, on the foundation quality assurance template, exactly what your organization considers the scope of quality assurance to be. If you have different types of quality assurance activities, use the Type field described in Section 4.2.8.1.

4.2.8.1 Type/Tier 3

As a simple example, different types of quality assurance activities may be:

- Desk Check
- Sample
- Review
- Inspection

In general, this list moves from activities that typically are more subjective to those that are more objective. Note that if you do not elect to use the Audit templates (discussed in Section 4.2.10) then you can capture audit activities as a fifth type of quality assurance activity.

4.2.9 ACTIVITY/CONFIGURATION MANAGEMENT TEMPLATES/TIER 3

Use this template to highlight and define those activities related to the configuration management process. As with other class-level templates, you may find it useful to distinguish different types of configuration management (as a function of "what" is going through the configuration management process).

4.2.9.1 Type/Tier 3

Some organizations have different types of configuration management. Often this is a function of the product. For instance, the configuration management process for plans may be fundamentally different from that for noncompliant products. Unless you essentially do only one type of CM, you should distinguish these activities by type. Examples include:

- Plan Control
- Design Control
- Product Control
- Noncompliant Product Control
- Design Control

4.2.10 ACTIVITY/AUDIT TEMPLATES/TIER 3

If you need to highlight audit activities, use this template. Be careful that you clearly distinguish why some activities are quality assurance, and other activities are audits. Typically, the term “audit” indicates an investigation by an outside or totally independent entity.

4.2.11 ACTIVITY/PROCESS IMPROVEMENT TEMPLATES/TIER 3

These templates can be used to emphasize activities specifically related to process improvement. Although not shown here, you may also want to include a “motivation” field for indicating whether this activity is CMM motivated, Malcolm Baldrige motivated, ISO 9000 motivated, internally motivated, etc. Allow for the fact that a given activity may be motivated by two or more quality initiatives.

4.2.11.1 Type/Tier 3

Process improvement is often conducted according to a relatively standard sequence of life-cycle phases. You may find it useful to distinguish process improvement activities as a function of which phase that activity supports. Example phases include:

- Process Improvement Process Research
- Assessment for Process Improvement
- Planning Process Improvement
- Piloting Process Improvement
- Technology Transfer
- Monitoring Process Improvement

4.2.12 ACTIVITY/TRAINING TEMPLATES/TIER 3

Use this class of templates to define activities related to training. Note that the results of these activities are often “intangible.” You can model the output of training by using the Product/Intangible/Skill template, discussed below.

4.2.13 ACTIVITY/PURCHASING TEMPLATES/TIER 3

Use this template whenever purchased products or services are an important part of your process.

4.2.14 ACTIVITY/SERVICE TEMPLATES/TIER 3

This template is for service that you provide to others, **not** service they provide to you. Most commonly, you use this template to capture how you service one or more products produced by the process.

4.2.15 ACTIVITY/DOCUMENTATION TEMPLATES/TIER 3

Often, the development of a product entails the creation and delivery of a considerable amount of documentation. Use this template to highlight activities dedicated to documentation development.

4.2.16 ACTIVITY/MEASUREMENT TEMPLATES/TIER 3

Measurement templates are used to show where, when, and how metrics are collected within your process.

4.2.16.1 Type/Tier 3

- Product
- Process
- General

Typically, you can divide your metrics into those that measure product characteristics and those that measure process characteristics. If you find it difficult to classify a metric into these two distinctions, then indicate it as a general metric. For the purposes of this template, there is no distinction between “measures” and “metrics.” Either or both can be documented using this template.

4.2.16.2 Collection/Tier 3

You will likely want to show measurement activities as collecting more than one measure or metric. This collection of fields repeats for each different measure or metric you collect.

- Metric ID (multiple occurrence)
- Unit of Measure

- Method of Collection
- Frequency of Collection
- Collected By

The primary purpose of this group of fields is to ensure consistency in how and how often metrics are collected.

4.2.16.3 Recorded In/Tier 3

Use this field to show where the metric information is stored. This may be a report, database, etc. Your process should also show where, when, and how the metric information is used.

4.2.17 PRODUCT TEMPLATES META-CLASS

As discussed in Volume 1, Section 3, product templates are used to show both what is needed by and used by the various activities.

4.2.17.1 Quality Rating Levels (Group of Fields)

- Quality Level/Unique ID (multiple occurrence)
- Quality Level Description
- Current Product Level of Quality: Tier 4

The Current Product Level of Quality field uses the same levels as described in Product Level of Quality Needed (discussed in Section 4.2.2.5). This field permits distinguishing between what is needed and what exists.

4.2.17.2 Product State Information (Group of Fields)

Most of the fields in this group are discussed in Volume 1, Section 3. However, at Tiers 3 and 4 you may find the following additional fields useful.

- *State System Identifier: Tier 3 (multiple occurrence).* As with activity states, State System Identifier is the unique identifier for a set of states. Any product may be subject to two or more systems of state transitions. As discussed in Section 4.2.2.9, complexity in a process model invariably introduces risk, and should you only create multiple state systems in those instances when doing so simplifies your model.
- *Label: Tier 1 (multiple occurrence).* Label was discussed in Section 3 of Volume 1.
- *Description: Tier 1.* Description was discussed in Section 3 of Volume 1.
- *Application: Tier 3.* When you define one or more systems of states for a product, any one set, or state system, may apply only to that template, or may also apply to all the children of that template. The Application field reflects this scope through the following two values:

- Node Only
- All Children
- **Augments: Tier 3.** Augments is a Yes/No field that is only relevant when a set of states has been inherited and when different state information is available at this level. A value of No in this field indicates the local state information completely replaces any inherited information. A value of Yes indicates that the local set of states extends or augments the inherited set.
- **Current Product State(s): Tier 4.** Current Product State(s) indicates, in an ongoing enactment of the process, the current state of this particular product. Note that if you have defined multiple state systems, then this would be a set of fields, one field for each state system. (Example states for a single state system were presented in Volume 1, Section 3.)

4.2.17.3 Related Activities (Multiple Occurrence)

This group was discussed in its entirety in Volume 1, Section 3.

4.2.17.4 Risks (Group of Fields)

This set of fields is exactly as discussed in Section 4.2.2.7, with the following two exceptions. First, the risks in this section are product-related risks, as opposed to activity-related risks. Consequently, the values appropriate for the Applies To field are different.

The fields that constitute this group are:

- Unique ID: Tier2 (multiple occurrence)
- Maximum Risk Tolerance: Tier 2
- Applies To: Tier 2
- Elaboration Text: Tier 2
- Risk Likelihood: Tier 3
- Risk Severity: Tier 3
- Risk Frequency: Tier 3
- Current Risk Likelihood: Tier 4
- Current Risk Severity: Tier 4
- Current Risk Frequency: Tier 4
- Delta Below Maximum Tolerance: Tier 4

As mentioned, these fields are used in a manner consistent with the explanation in Section 4.2.2.7. However, Applies To, for product risks, takes one of the following values:

4.2.18.1 Type/Tier 3

- Developed Product
- Value-added Product
- Purchaser Supplied Product
- Measurement Product
- Plans
- Policies and Procedures
- Standards

Product types are highly dependent on your business domain, customers, market, etc. Tailor the list of types to suit your specific circumstances.

4.2.18.2 Security Classification/Tier 2

Products from your process may need to be characterized by a security designation. Examples of designations include public, company proprietary, secret, top secret, classified, etc. In an elaborate enactment or simulation model, you will want to ensure that the roles or people assigned to an activity have the appropriate clearance levels to access the products provided to or generated by that activity.

4.2.19 PRODUCT/TANGIBLE PRODUCT/CONTRACT DELIVERABLE TEMPLATE

One of the most important product distinctions you can make is whether a given product will be delivered to a customer or not. If so, use this template for your detailed product information.

4.2.20 PRODUCT/TANGIBLE PRODUCT/CONTRACT NONDELIVERABLE TEMPLATE

If a product is not intended for delivery to a customer or client, but is required by the contract, use this template. If you do not need to highlight the distinction between delivered and nondelivered products, you can use the Destination field at the meta-class level to capture this distinction. Otherwise, your selection of this subclass template must be consistent with the value of the Destination field.

4.2.21 PRODUCT/TANGIBLE PRODUCT/NonCONTRACT NONDELIVERABLE TEMPLATE

Use this template to represent any products generated or used during the process that are not required by the contract and not delivered to the customer. For example, your process might require that you design and develop custom software to test and evaluate the performance of products required by the contract. If the contract does not require you build that custom software and you do not intend to release it to the customer, you represent it with this template.

4.2.22 PRODUCT/TANGIBLE PRODUCT/STATUS REPORTS

If status reports do not require high visibility within your process, you do not need this template. Instead, you can simply add "status reports" as one of the tangible product types listed in Section 4.2.18.1.

However, there are a variety of reasons where you might want to explicitly highlight the generation and use of status reports within your process. For example, you can use status reports as the critical mechanism that controls your process. In such a model, status reports are used to convey the states of products and activities, product quality, etc. An activity would then look to various status reports to determine whether it can execute or continue executing, and may itself generate one or more status reports during execution or at completion.

If you use this template, and if some status reports are required as contract deliverable, and others are noncontract, nondeliverable, you will need a Type field on this template to capture those distinctions.

4.2.23 PRODUCT/INTANGIBLE PRODUCT TEMPLATE/TIER 2

Aside from tangible products, there are also intangible products. Intangible products result from activities occurring that yield value to you or your customer, but which do not necessarily produce something tangible. The two primary examples of intangible products are knowledge and skills.

4.2.24 PRODUCT/INTANGIBLE PRODUCT/KNOWLEDGE TEMPLATES/TIER 3

When an activity involves the accumulation of important information, but does not yet result in any documentation of that information, use the knowledge template to show how this “product” (knowledge) is both generated and used.

4.2.25 PRODUCT/INTANGIBLE PRODUCT/SKILL TEMPLATES/TIER 3

Skill templates are one of the most common “output products” of training activities.

4.2.25.1 Proficiency Rating Levels (Group of Fields)

In addition to the efficiency fields inherited from the support template, role templates (Section 4.2.27) can also show roles at different levels of proficiency. Proficiency ratings on this skill template indicate the different levels of ability you may need for a role to have at various times or places within your process.

This group of fields consists of:

- *Proficiency Level/Unique ID (multiple occurrence)*. Each Proficiency Level needs a unique identifier. Use the following scale:
 - Academic
 - Trained
 - Novice
 - Experienced

- Highly Experienced
- Expert

Role templates will use these values to indicate how proficient a role needs to be at this and other skills.

- **Proficiency Level Description.** Use the Proficiency Level Description field to explain the distinguishing characteristics of each level of proficiency.

4.2.26 SUPPORT TEMPLATES META-CLASS

Support templates include both roles and resources. Type of access, sharing, locks, etc., all become important at higher tiers of usage.

4.2.26.1 Type of Access Available/Tier 3

For both roles and resources, it is necessary to know whether the support is only available in a shared capacity, only available in an exclusive capacity, or both. Use the following field values to capture this information:

- Shared Only
- Exclusive Only
- Shared and Exclusive

4.2.26.2 Share (Group of Fields)

If a support is available in any type of a shared capacity, you need additional information. Use this group of fields to capture information on how sharing occurs:

- **Share Limit: Tier 3.** If a role, resource, or tool can be shared, it is sometimes the case it cannot be infinitely shared. Use the Share Limit field to put a limit on how many activities the support can be shared between.
- **Current Share Count: Tier 4.** The Current Share Count field is used to hold an actual count of the current number of "shared locks" on this role or resource. When Current Share Count equals Share Limit, then the support can not participate in any other shared support activities (until one or more activities that currently hold locks release them).

4.2.26.3 Security Clearance/Tier 2

If your process involves products that carry a security classification (see Section 4.2.18.2) then the Supports for your process may likewise need to be characterized by a security clearance. (Example clearances include none, company proprietary, secret, top secret, classified, etc.) Note that the set of security levels used to characterize your products and supports needs to be consistent.

In addition to people having sufficient clearance to access a sensitive product, you might also need to apply this concept to other supports. That is, it may be that only certain copy machines may be used

to duplicate classified material, that only particular rooms can be used when reviewing or discussing top secret information, that disposal of secret material must only occur in certain rooms or containers, etc.

4.2.26.4 Supported Activities (Multiple Occurrence)

Most of the fields in this group were discussed in Volume 1, Section 3. In addition to that information, for every activity needing this support, it is necessary (at Tier 4) to know what type of “lock” is held on the support. Generally, at any given moment a support will have: (a) no locks upon it, (b) one exclusive lock and no other locks, or (c) up to “share limit” shared locks. Additionally, at any moment there may be an arbitrary number of requests for locks pending from the various activities.

In addition to the Tier 1 fields, this group includes Current Lock Status: Tier 4.

For every activity that has or needs a lock on the product, the type of lock (or request) is added to the other information about that activity. The values for the Current Lock Status field are:

- Shared Lock Held
- Exclusive Lock Held
- Requesting Shared
- Requesting Exclusive

4.2.26.5 Efficiency Rating (Group of Fields)/Tier 4

Efficiency Rating for supports allows you to explore the implications of using more or less efficient roles or resources in supporting the various activities. This group of fields consists of:

- *Efficiency Level/Unique ID (multiple occurrence).* Each level of efficiency needs an Efficiency Level/Unique ID. You should use a three- or five-step scale that ranges from (well-)below average to (well-)above average.
- *Efficiency Level Description.* Use the Efficiency Level Description field to explain the distinguishing characteristics of this level of efficiency.
- *Efficiency Impact Multiplier.* Each activity has an average duration. However, you will want to make the estimated duration of an activity a function of the efficiency of the roles, tools, and resources assigned to that activity. An Efficiency Impact Multiplier of 1.0 has no impact on durations. Multipliers greater than 1 will result in longer durations. Multipliers less than 1, shorter durations.

4.2.26.6 Current Level of Support Efficiency/Tier 4

During dynamic simulation or enactment you may want to vary the efficiency of a support or otherwise tie the efficiency to a variety of other factors (e.g., reduce level of efficiency with each new “shared lock” on the support). Use this field to show the actual (as opposed to default) efficiency level. Although you can establish this field as a direct multiplier, it is recommended that you use an identifier from one of the efficiency levels in the Efficiency Rating group of fields instead (Section 4.2.26.5).

4.2.26.7 Risks (Group of Fields)

Risks associated to supports are similar, in how you define them, to risks associated with products or risks associated with activities. Note that the following fields are identical to those named in the risk groups discussed in Section 4.2.2.7 (activity risks) and Section 4.2.17.4 (product risks). The only exception is that this group of fields does **not** contain the field Applies To. When you associate a risk with a support, it applies to that support in general.

- Unique ID: Tier 2 (multiple occurrence)
- Maximum Risk Tolerance: Tier 2
- Elaboration Text: Tier 2
- Risk Likelihood: Tier 3
- Risk Severity: Tier 3
- Risk Frequency: Tier 3
- Current Risk Likelihood: Tier 4
- Current Risk Severity: Tier 4
- Current Risk Frequency: Tier 4
- Delta Below Maximum Tolerance: Tier 4

An example of support risk is Unavailability. For instance, if you expect to normally **not** have a role on staff, but plan a process that presumes you can acquire the needed skills through hiring, then you may want to highlight that risk:assumption.

4.2.26.8 Support State Information (Group of Fields)

For advanced applications, you will find it useful to expand on the state information for supports and how to interpret that information, by adding the following fields:

- State System Identifier: Tier 3 (multiple occurrence)
- Label: Tier 3 (multiple occurrence)
- Description: Tier 3
- Application: Tier 3
- Augments: Tier 3
- Current State(s): Tier 4

The use of these support state fields is similar to the descriptions provided in Sections 4.2.2.9 (activity states) and 4.2.17.2 (product states).

4.2.27 SUPPORT/ROLE TEMPLATES

Role templates are used to show how humans are involved in the process. With roles, it is important to capture information about reporting structures; skill proficiencies; authority; and, in highly advanced applications, who within your organization is qualified to hold which roles.

4.2.27.1 Reports To/Tier 2

Discussed in Volume 1, Section 3.

4.2.27.2 Reported To By/Tier 2

Discussed in Volume 1, Section 3.

4.2.27.3 Unique Individual/Tier 4

This is a Yes/No field that indicates whether the template is being used to represent a specific and real individual within the organization. If this is truly a role, and not a real person, the field value is No, otherwise, use Yes.

4.2.27.4 Proficiency (Group of Fields)/Tier 3

Skill templates (a subclass of intangible products) are used to show skill as an output product of training. Roles need certain skills. Use this group of fields to show the skill levels necessary for this role. The fields in this group are:

- ***Required Skill Area Unique ID (multiple occurrence).*** Required Skill Area Unique ID is the unique identifier of a skill template.
- ***Minimum Level of Proficiency.*** The skill template defines a group of fields (Section 4.2.25.1) which define the different levels of proficiency, or expertise, that apply to the skill. Select one of those levels of proficiency to represent the Minimum Level of Proficiency that must be achieved to have adequate skill in this area.

Note that if this role template is being used to the level of detail where it represents an actual person (that is, Unique Individual field (Section 4.2.27.3) has a value of Yes), then the semantics of this field is not “minimum” level of proficiency, but is instead “actual” level of proficiency.

- ***Required Skill Area Elaboration Text.*** Use the Required Skill Area Elaboration Text field to provide a text explanation about the use or importance of the skill to the role.

4.2.27.5 Internal Constraint Waiver Authority/Tier 3 (Group of Fields)

If you use internal constraint templates, the implication is that some role exists which has the authority to waive the constraint.

- ***Template Unique ID (multiple occurrence).*** Use the Template Unique ID field to list the identifier(s) of internal constraint templates for which this role has waiver authority.

- **Waiver Elaboration Text.** Use the Waiver Elaboration Text field to provide any details or additional information about the nature of or conditions surrounding when, why, or how a waiver occurs.

4.2.27.6 Approved Staff Unique ID (Multiple Occurrence)/Tier 4

If this template represents a role, as opposed to an actual person (see Section 4.2.27.3), then list all the identifiers of actual persons approved to hold this role. When the role template represents a real person, this field is meaningless and left blank.

4.2.28 SUPPORT/TOOL TEMPLATES/TIER 2

Use the tool template to specifically highlight the primary tools that participate in the process. Note that if a tool involves a network site license which limits simultaneous use to, for instance, 8 people, then you will want to set the field Share Limit to 8 (see Section 4.2.26.2).

4.2.29 SUPPORT/RESOURCE TEMPLATES

Use this template to represent any resource needed to perform the process. This includes training rooms, meeting rooms, various office supplies, etc. Needless to say, only represent those resources considered important in either understanding, analyzing, or performing the process. This template only adds one field to those inherited.

4.2.29.1 Owned By/Tier 3

This field indicates ownership of resources. Whether you own a particular resource can significantly impact the types of risk you have with respect to that resource.

4.2.30 CONSTRAINT TEMPLATES META-CLASS/TIER 2

Constraint templates have two important uses. First, use these templates to indicate any constraining influence that is not better represented as an activity, product, or support. Second, use constraint templates to simplify your models. For instance, if you have 30 different entry criteria for a particular activity, you can clean your model by showing, for instance, a constraint that indicates "Task Manager decides activity may start."

These tradeoffs always involve a price. On one side you have detailed accuracy, but may lack clarity. On the other side, clarity is improved, but details are lost. At lower tiers, opt for improved clarity; at higher tiers, you will need the details.

If you find yourself using a high number of constraint templates (maybe, higher than 10% of your total templates) then examine the constraints and see if you can extract another meta-class. If so, your model will be more informative when you establish that meta-class and elaborate it with appropriate details.

You may have constraints on activities, products, and supports. Generally, however, if a constraint is applicable to products (for instance), it is only applicable to products. This is not a rule, it is just in the nature of constraints. If you have a constraint that applies to more than one meta-class, reexamine that constraint to be sure that you are not inadvertently using it to represent two different things.

Additionally, the nature of the constraint may introduce risk.

4.2.30.1 Constrained Activities (Multiple Occurrence)/TIER 2

When the constraint applies to one or more activities, put the activity template unique identifiers in this field.

4.2.30.2 Constrained Products (Multiple Occurrence)/TIER 2

Use this field for unique identifiers of products subject to this constraint.

4.2.30.3 Constrained Supports (Multiple Occurrence)/TIER 2

If the constraint applies to roles, tools, resources, or other supports, put the support template identifiers in this field.

4.2.30.4 Constraint Induced Risks (Group of Fields)/TIER 2

The fields in this group are identical to those discussed in Section 4.2.26.7 (support risks). Because anything that influences your process can be modeled as a constraint, you may also find it useful to include the Applies To field to this group of fields. Examples of its use can be found in Section 4.2.2.7 (activity risks) and Section 4.2.17.4 (product risks).

4.2.31 CONSTRAINT/INTERNAL CONSTRAINT TEMPLATE

Internal constraint templates are used to represent any constraint where those who participate in the process have the option to potentially receive a waiver to that constraint.

4.2.31.1 Waiver Criteria/TIER 2

Use this field to describe, in detail, the criteria that justifies a waiver to this constraint.

4.2.31.2 Roles Authorized to Exercise Waiver (Multiple Occurrence)/Tier 2

The role templates show which internal constraints a role has waiver authority over. This field represents the opposite side of that view. That is, use this field to show which roles have waiver authority over this constraint. Your environment may automatically maintain this type of "reverse relation" and, if so, you do not need this field.

4.2.32 CONSTRAINT/EXTERNAL CONSTRAINT TEMPLATE

External constraints are constraints for which there are no waivers. An example of this might be your organization's safety policies or standards.

4.2.33 RISK TEMPLATE META-CLASS/TIER 2

The last meta-class of templates is used to represent risk. Each of the other meta-classes (activities, products, supports, and constraints) all have fields to reflect the types of risk that may be associated with them. Use risk templates to capture details about these risks and their degrees of impact.

4.2.33.1 Severity Levels (Group of Fields)/Tier 2

Each risk should be divided into several levels, where each level indicates severity of impact. At a minimum, show two levels of risk (Moderate and Severe). At the other end of the spectrum, avoid using more than a seven level table (ranging from Extremely Low to Extremely Severe). This group of fields consists of:

- Level/ID Number (multiple occurrence)
- Level Description

Use the Description field to explain how the severity of the consequences can be distinguished at each level.

4.2.33.2 Activity Risks Unique IDs (Multiple Occurrence)/TIER 2

This is a “reverse relation” field that you may not need, depending on the automated environment you are using. For any activity template that indicates the activity is subject to this risk, show the activity’s unique identifiers in this field.

4.2.33.3 Product Risks Unique IDs (Multiple Occurrence)/TIER 2

This too is a “reverse relation” field. For any product template that indicates the product is subject to this risk, show the product unique identifiers in this field.

4.2.33.4 Support Risks Unique IDs (Multiple Occurrence)/TIER 2

Use this field to establish the reverse relation from risks to support templates.

4.2.33.5 Constraint Risks Unique IDs (Multiple Occurrence)/TIER 3

This field contains identifiers for constraint templates that reference this risk template.

4.2.34 RISK/COST RISK TEMPLATE

Use this template whenever the primary impact of a risk is cost.

4.2.35 RISK/QUALITY RISK TEMPLATE

Use this template to capture information about quality risk. If you use the performance and construction risk templates, then exclude those characteristics from consideration as quality attributes. If you elect not to use performance and construction risk templates, then consider those factors as relating to product quality and use this template.

4.2.35.1 Type of Quality Risk/Tier 3

At a minimum, quality risk can be divided into that which applies to products and that which applies to the process producing the products. In the event that a risk does not seem to clearly apply either to products or activities, show it as a General risk.

- Product
- Process
- General

Typically, product risk will be mapped only to product templates, process risks will be mapped only to activities, etc. However, there may be exceptions. When you encounter an exception, take a moment to verify that you are accurately relating risks to the appropriate process objects.

4.2.36 RISK/SCHEDULE RISK TEMPLATE

Sometimes the primary consequence of a risk is its impact on schedule. Use this template to define these risks.

4.2.37 RISK/PERFORMANCE RISK TEMPLATE

The performance risk template is used to define product performance risk and process performance risk.

4.2.37.1 Performance Risk Applies To/Tier 3

As with the quality risk templates, this field can have one of three values:

- Product
- Process
- General

When the speed, throughput, or other performance characteristics of a product represent an important risk consideration, it is product risk; when the risk relates to how fast or efficiently activities are performed, show it as a process type of risk.

4.2.38 RISK/CONSTRUCTION RISK TEMPLATE

This risk applies entirely to products. When you think that the quality of construction (as opposed to how a product performs) is at risk, use this template.

4.2.39 RISK/PREDICTABILITY RISK TEMPLATE

You will note, as you analyze and perform risk management, that there are certain common relationships that affect the risk tradeoffs you make and their impact on your overall risk exposure. For example, by increasing the number of quality reviews within a process, you usually reduce your quality risk, but increase your cost risk and increase your schedule risk. Conversely, by eliminating some of the intermediate product inspections and tests, you might reduce your schedule risk, reduce your cost risk, but significantly increase your quality risk.

Occasionally, you may elect to eliminate activities from a process in a way that, at first glance, seems to reduce cost risk, schedule risk, and quality risk, and have no impact on performance risk or

construction risk. Such “no-cost” improvements are exceedingly rare and should be viewed suspiciously. Often, what you have lost is “predictability” in the outcome of your process. Uncertainty in a process is undesirable. When actions (or the lack of actions) adversely impact your confidence in meeting project plans, you can represent this using predictability risk templates.

4.3 SUMMARY

The more advanced your usage of process information, the greater the likelihood that information, and its relationship to other information, will be unique to your circumstances and needs. At Tier 1, the fields are generally applicable to all circumstances, environments, and business domains. However, by the time you arrive at Tier 4 usage, the data you collect, its relative importance, and how you use it will all depend extensively on your budget, the experience of your people, your organization’s tactical and strategic goals, the relative dependence of those goals on process definitions and models, and various other process drivers.

If one of your organizational, business area, or program process goals is to export process information from your repository and reformat it into a different representation, you will want to verify that the information you collect accommodates the needs of that alternative representation.

A brief overview of alternative representations is presented in the next section. It discusses their general characteristics, uses, and the type of information necessary to support them.

This page intentionally left blank.

5. ALTERNATIVE PROCESS REPRESENTATIONS

5.1 CHARACTERISTICS OF PROCESS REPRESENTATION NOTATIONS

When comparing alternative process notations and their accompanying methodologies, relative advantages and disadvantages can be highlighted by contrasting the various approaches from the perspective of key characteristics. Key characteristics that determine the suitability (and hence, the subjective value) of different process notations include:

- Scalability
- Applicability
- Flexibility
- Readability
- Maintainability
- Learnability
- Robustness
- Relative formality
- Representative power

5.1.1 SCALABILITY

Scalability represents the degree to which a notation can simultaneously tolerate both abstraction and details. Some notations may be especially good at representing high-level abstractions of a process, but rather inadequate at capturing process details. Other notations may excel at representing details, but may suffer severe "scale-up" problems when it comes to representing high-level views of the process. An ideal characteristic for a notation is to efficiently provide both abstract views and detailed views of a process.

5.1.2 APPLICABILITY

A notation can be more or less applicable to the specific needs of a given domain of processes. For example, some notations may be quite poor at representing parallelism; others may be quite poor at representing the people involved in a process; and others may provide only a static view of a process

but no insights into dynamic characteristics. If, for example, parallelism, people, or process dynamics are important to the domain of processes being represented, it is ideal to select a notation that can capture and represent these and similar details important for and applicable to your specific domain.

5.1.3 FLEXIBILITY

Flexibility is the degree to which a notation can be altered and customized to better achieve specific objectives. Note that this is a different quality than maintainability (described in Section 5.1.5). Flexibility is the characteristic of being adaptable outside of a specific domain. A notation is relatively flexible to the degree that it can be tailored for your specific needs.

5.1.4 READABILITY

A process notation is relatively readable to the degree that it allows a reader to quickly, clearly, and accurately derive insights into the described process. An additional consideration with readability is the amount of training or expertise required before a notation becomes readable. Complex mathematical expressions may require years of relevant education before they can be accurately interpreted; whereas, a simply flowchart type approach may convey meaningful information even to the uninitiated.

5.1.5 MAINTAINABILITY

A process notation is more or less maintainable as a function of the ease with which changes can be made to existing representations. Process improvement implies process change. Any process definitions or models that you use to document the existing process will, therefore, need regular updates. The ease with which such updates can be done is a direct reflection of the maintainability of that notation. Although maintainability is affected by, for example, readability, flexibility, and learnability, it is a separate consideration. Just because something is readable, flexible, and learnable does not guarantee that it is maintainable.

5.1.6 LEARNABILITY

The learnability of a process notation is derived from three factors:

- The average level of expertise needed by individuals before they can receive training in using the process notation
- The average amount of training required before they are capable of using the notation
- The average amount of time it requires a trained person to transition from just being capable to actually being proficient and adept at using that notation

A highly learnable notation requires little or no related expertise, perhaps a day or two of training, and it allows process engineers to become proficient in a few weeks or months of actual usage.

5.1.7 ROBUSTNESS

A process notation is robust if it can be used "as is" on a comparatively larger set of different processes. Note that higher robustness can, to some degree, compensate for a lack of flexibility. There is less need

for a notation to be tailorable for different uses when that notation is usable in standard form across a wide variety of applications. Likewise, a low level of robustness in a notation can be offset by a relatively high degree of flexibility.

5.1.8 RELATIVE FORMALITY

The formality of a notation is a combination of a variety of factors. At a minimum, relative formality is reflected by the degree to which a notation is unambiguous and deterministic. Formality increases to the degree that it constrains the set of elements with which processes can be described, and it defines the operations and transformations permissible on those elements. An ideal level of formality for a process notation is one that allows process representations or models to be machine compilable, linkable, and executable. Even more formal are "provable" notations where it can be proven (using mathematical techniques) that the design representation is consistent with and complete against an analysis representation, and the enactment representation is consistent with and complete against the design.

5.1.9 REPRESENTATIVE POWER

How broad a spectrum of representational techniques does the notation contain? For example, some so-called broad spectrum programming languages combine a wide array of logic, functional, procedural, parallel techniques. As a process definer, you may be interested in how a notation represents procedure, rules, and data or, from another perspective, function, behavior, and organization.

Of interest as well are its "structuring" capabilities reflecting your concern with factors such as granularity, redundancy, modularity and information hiding. You may be interested in its compatibility with an object-oriented approach.

Finally, the notation's relationship with and power in describing and generating various media, forms (e.g., guidebooks, tutorials, training charts), analytical and simulation models, and automated process management or enactment may be of interest.

5.1.10 CHARACTERISTICS OF "BEST" REPRESENTATIONS

The characteristics introduced in Section 5.1 are not mutually exclusive and, to greater or lesser degrees, each one overlaps the others. Nevertheless, these are key factors that you need to consider when matching the level of process definition and representation with your situation and strategy.

The "best" notation for you will be driven by a variety of factors that not only differ between organizations but which also change, in time, within the same organization. Regardless of your choice, it is clear that some representation beyond free-text based descriptions is critical for documenting, analyzing, and communicating about your process.

You are likely to benefit from an approach that is highly dynamic; makes change management an explicit, accepted, and deliberate part of the process; and is flexible, maintainable, and easily tailorable. As discussed in Section 5.5, these are all deliberate characteristics of MPDM.

Additionally, you may also want to represent your process information using different styles, notations, graphical conventions, etc. Alternative techniques for representing your process are discussed in Section 5.4.

5.2. DEGREES OF FORMALITY

The most common technique for representing or defining processes is also the least formal: descriptive text. Essentially, all operations manuals are text-based representations that describe one or more processes. Organizational policy and procedure manuals are also potential sources of process descriptions. One significant advantage to text-based descriptions is that there are virtually no constraints placed on the description itself. Since the notation is the entire language of words, anything that can be talked about, can be described. This, however, is also the most significant disadvantage to text-based process descriptions. Since there are no constraints (other than grammar) on the structure of the description, there is the potential for considerable inconsistency, ambiguity, uncertainty, and inaccuracy. Text-based descriptions are typically the least formal type of process representation.

NOTE: A text-based process description is distinctly different from an information-model-based description that produces a text-intensive end-product (described extensively in Volume 1.)

At the other end of the formality spectrum is mathematics. Although mathematically provable process models are an ideal goal, such methodologies are still an area of active research, and there are not any clearly cost-effective and widely applicable math-based methodologies. However, between the informality of descriptive text and the rigorous formality of mathematics, there are many options for representing processes at varying degrees of formality.

Having graphical support to a notation can directly contribute to higher degrees of formality. Usually, a graphically-oriented notation comes coupled with a methodology that imposes rules on placement, use, and connections between the graphical objects. These rules constrain the types of structures that can be built and increase the formality of the resulting depictions. Additionally, graphical depictions are especially useful for portraying abstract or high-level relationships. The resulting diagrams allow for insights into the process that can be virtually impossible to derive from descriptive text.

Fortunately, there are enough parallels between the need to represent human or organizational processes and the need to represent software-based processes that many of the techniques and notations developed for use in software engineering can be extended for use in general process representation. From a software perspective, any software program can be written using five basic constructs:

- Sequence
- Selection
- Iteration
- Dispatch
- Rendezvous

These are exactly the same constructs needed to represent organizational processes. Any type of process can be represented if you have:

- A means for indicating a series of activities occurring in sequence
- A means for selecting between two or more optional activities

- The repetitive execution of one or more activities
- The parallel initiation of two or more activities
- The synchronization of two or more activities executing in parallel

Examples of software-based techniques that can be used in general process modeling include State Transition Diagrams (STDs), ETVX, SADT, Statecharts, and Petri nets. These and similar alternative notations will be examined later in Section 5.4.

5.3. CHOOSING THE RIGHT PROCESS REPRESENTATION NOTATION

When deciding how to initiate a program of process representation, the selection of a process notation is of central importance—it is the language you will be using to communicate about your processes. The key characteristics described in Section 5.1 are important for guiding your thinking in choosing a process representation notation. However, primary considerations when choosing an approach to process representation depend upon your relative comparisons of characteristics that you consider to be important.

Remember that all notations imply some type of methodology by which the notation is used. When considering alternative process notations, you need to simultaneously consider the associated methodology. Both the notation and its accompanying method of use determine its relative scalability, readability, and robustness.

Some of the questions you will need to answer include:

- Can the notation represent details and also abstractions? (Scalability)
- Can the notation be used to represent your processes? (Applicability)
- Can the notation be adapted for use in representing different processes? (Flexibility)
- Are process descriptions resulting from this notation easily interpreted? (Readability)
- Can process depictions be easily updated to reflect changes? (Maintainability)
- Can average people become quickly competent without extensive training? (Learnability)
- Is the notation capable of representing a large variety of processes? (Robustness)
- Does the notation yield machine-interpretable process models? (Formality)
- How many different types of products can be supported? (Representative Power)

There are no simple answers to these questions. In all cases, the answers depend on several important factors relevant to your organization. As discussed in Sections 5.3.1 through 5.3.5, these factors include the type of environment you have, the resources available, budget constraints, history, and your immediate and future goals.

5.3.1 ENVIRONMENTS

Environments vary significantly between organizations, and they can even vary significantly between divisions within a single organization. From the perspective of process representation, one major

consideration is the relative volatility of your environment. Are the processes you intend to model relatively stable or highly dynamic? Are the changes relatively nominal, or are they often radical? Are the changes predictable, or are they often unexpected? Additionally, is the environment dedicated to a single domain of processes, or are there processes occurring within a variety of domains? Are the processes of relatively short duration, or are they long-term? Are the processes highly iterative, or typically nonrepeating? Is management largely a human-intensive activity, or is there a large degree of automated project management support? These are examples of environmental considerations that can strongly influence which process notational characteristics are applicable to your specific site.

5.3.2 RESOURCES

Another consideration is the resources you have available to perform process analysis, design, and representation. From the perspective of people, do you have highly trained or experienced process engineers already available or will you need to have people trained? Are you anticipating a large initial effort (involving dozens of people), or will you initially involve only a few people? Will your process engineers be full-time, dedicated resources; or will process-related work be a matrixed responsibility added to their current responsibilities? Are you intending to use the process representations to support project management in an automated, integrated environment? If so, do you currently have the tools and techniques needed to support that approach, or will you have to acquire them?

5.3.3 BUDGET CONSTRAINTS

Budget constraints are another major factor. How much funding is available to you for initiating the process representation effort? After start-up costs, how much funding is available to support the ongoing program? Is immediate cost justification important, or can the costs be amortized over a longer period of time?

5.3.4 HISTORY

Organizational history must also be considered. Is there an existing repository of process representations? If so, are they rendered in one or multiple notations? Is there an existing experience pool for the performance of process engineering? Was process engineering an effort previously done but canceled long ago or has it been an ongoing effort? Is the staff allocated to process engineering being held relatively constant, or is it being significantly increased or decreased?

5.3.5 GOALS

Finally, and most importantly, you must consider both the organization's immediate and long-term goals. It is important to take a goal-driven approach to process representation. "Improved process maturity" is too ambiguous a goal to help answer many of the questions presented here. Instead, explicit and well-defined goals are needed. Is the goal to develop a process guidebook? If so, is the guidebook intended to be a technical reference or a self-teaching tutorial? Is the goal to develop a guidebook of a **proposed** process that can then be distributed to reviewers and process analysts? Is one of the goals to develop material that will facilitate training employees in organization processes? Is one of the goals to have a representation that facilitates gaining insights into dynamic and static process characteristics? Does the representation need to facilitate developing automated process models that can be executed? Is the goal of process model execution to gain insights into performance

issues such as deadlocks and bottlenecks? Is it a goal to use the representations to facilitate project management?

5.3.6 ORGANIZATIONAL ISSUES AND PROCESS CHARACTERISTICS

The answers to questions in Section 5.3.5 vary from one organization to another; and even within the same organization the answers will change with time. Nevertheless, there is a general mapping that can be made between the key characteristics of alternative notations and the organizational issues presented in Section 5.3. To reiterate, the issues are:

- Your environment
- Available resources
- Budget constraints
- History
- Your immediate and future goals

If your environment is volatile, then a notation with a high degree of maintainability is desirable. Ideally, as your process goes through significant change, the notation and methodology you have chosen will remain usable. If you will be defining and modeling processes across multiple domains, robustness and flexibility become important. You will need a notation that is either capable of handling a variety of processes or one that can be easily adapted to your changing needs. Conversely, if you will be targeting a single domain, applicability is of key importance. Long-term processes imply a need for a notation that exhibits a high degree of maintainability; and highly integrated, automated environments may convey a need for greater formality in the selected notation.

The type of available resources also affects your choice of a notation. If you anticipate a high rate of turnover in the personnel performing as process engineers, then you will need a notation that has a high degree of readability and learnability. Conversely, if many of the process engineers are already familiar with a particular notation, then learnability is less important—your resources already have much of the fundamental training and experience.

If you must work within a very small or very tight budget, you will need a highly scalable notation. This gives you the opportunity to build either very high-level abstract representations or, if it is more expedient, to build very low-level detailed representations of isolated or highly cohesive sub-processes. Either way, a considerable amount of progress can be accomplished with comparatively little time and effort. Similarly, if you need short-term confirmation of beneficial results, then both scalability and learnability are crucial. Under such budget or schedule constraints, you cannot afford a notation that requires either extensive training or labor- and time-intensive application.

History is a key consideration, but so is your view of history. It may be that your organization has built a small resource pool of people familiar with the use of a particular notation. If the notation is applicable to your process domain, then it is sensible to give close consideration to that notation for representing your processes. However, it also may be time to depart from history and introduce a new notation. In short, the applicability, learnability, and maintainability of a notation can rapidly become more important to you than history of usage within your organization.

A notation's relative support of your current and future goals can involve several, if not all, of the key characteristics already discussed. If your goal is to develop process guidebooks or self-teaching tutorials, for example, then readability and learnability are primary considerations. If your goal is to develop process models and study them for temporal or behavioral characteristics, then you need a notation capable of capturing process dynamics. This implies you need a notation with a relatively high degree of formality. If one of your long range goals is integrated process automation, then formality may become of central importance.

As a general rule, the more flexible, robust, maintainable, and learnable a process representation approach can be, the more usable it is. As certain environmental, budget, resource, historical, or goal-oriented constraints become more important, others become less important. Nevertheless, there are advantages to having an approach to process definition and modeling that manifests all of these characteristics to a high degree. As discussed in Section 5.5, this is the rationale behind the template-based process representation approach proposed in this guidebook.

5.4. ALTERNATIVE MODELS AND REPRESENTATIONS

The MPDM templates can be used as an analytical tool to facilitate the organized collection of information that will later be used to develop process models in SADT, ETVX, etc. The fields on the templates are intentionally designed to capture a superset of information needed by alternative representations. From this perspective, if a process representation already exists in the format of an alternative notation (such as ETVX, SADT, etc.), the templates can be used to augment or extend that representation by including additional information and relations that the templates provide. The challenge facing you is to select which (if any, but it may be several) alternative notation you will need to use in the future.

Alternative approaches to process modeling can be contrasted by examining the objects elaborated by each approach and the relationships permitted among those objects. There is no universal standard for what is a "better" approach and what is "worse." All that can be evaluated is which approach has the greatest potential for helping you achieve your goals, in your environment, using your resources, given the funds available in your budget, etc. Additionally, significant changes in any of these variables may render one approach progressively less beneficial and another approach more beneficial. Therefore, the purpose of this section is not to provide a rank-ordering of better and worse approaches to process definition and modeling; instead it is to provide a basic framework that you can use when performing your comparative analysis, and to present information that you can use in determining the approach best suited to your requirements.

Sections 5.4.1 through 5.4.7 discuss the following seven alternative process representation and modeling notations:

- State Transition Diagrams (STDs)
- Entry Task Verification eXit (ETVX)
- Structured Analysis and Design Technique (SADT)
- Statecharts
- Petri nets

- Process and Artifact State Transition Abstraction (PASTA)
- RINs (Role Interaction Nets)

These notations can accomplish a variety of objectives. STDs are discussed first because they offer one of the simplest approaches for modeling a process. Although STDs are quite limited when compared to other techniques, they can be quite useful for learning or teaching purposes.

ETVX and SADT are widely accepted by industry. Originally designed as techniques for representing software-based or software-oriented processes, they can be generalized for use in process definition and modeling.

Statecharts and Petri nets offer opportunities for gaining insights not only into the static but also the dynamic characteristics of a process. PASTA is included because it was designed to support environment-based process modeling.

Finally, RINs are included as an example of a notation that readily translates into Petri nets and/or Statecharts.

All these alternative process representation notations are compatible with the templates. Although the templates can be used as a standalone technique for representing a process, they can also be used prior to, in conjunction with, or subsequent to these alternative notations.

5.4.1 STATE TRANSITION DIAGRAMS

STDs are often used for describing finite automata (finite state machines). Any process that can be described in terms of a finite automaton can be represented using an STD. Generally, a finite automaton accepts some series of input symbols and typically produces some series of output symbols (Kollman and Busby 1984). Each such output symbol is a function of the relevant input symbol, the current state of the automaton, or both. Additionally, as input symbols are received by the automaton, its state may change. With regard to process modeling, input symbols and output symbols typically represent milestones that occur in reality, and states within the machine represent different phenomena or activities within an overall process. States have a time dimension, state transitions do not. Thus, finite state machines can be seen as one possible representation for modeling sequences of phenomena within some defined domain. When modeling the behavior of large or complex systems, it is often useful to discuss both states and sets of states. The term "mode" is optionally used to distinguish a state set (Sanden 1992a).

The dynamic behavior of a state transition machine is simply described. All state machines, at the outset, are set in their initial state. Whenever a legal input symbol is received, the machine always transitions to the next state, which may or may not be identical to the current state (this is the state transition).

From the data collection templates, it is state information captured by the templates that directly translates to STD models. It is often easier to represent events as arcs. The arcs among events can be throughput and state transitions; however, STDs are not good for representing supports and constraints. Also, you may want to consider an STD for throughputs (as they have state transitions also); but typically, not much insight would be gained from such a diagram (that is, it is likely to be just a chain of states, perhaps with a simple loop involved).

Because of scale-up problems, STDs are best used when a process (or part of a process) is not excessively large or complicated. Another advantage to STDs is that, due to their relative simplicity, the initial costs for training and the learning curve are likely to be lower than using other less familiar or more complex process models.

From the perspective of the objects and relationships used within the templates, STDs primarily capture activity-activity sequence relations. However, carefully constructed STDs can convey an intertwined model that not only shows activities and their states, but also products, and their evolution through a variety of states. Note that product state transitions can be used to drive the ordering of activities, activities can be used to determine the states of products, or some combination of these two approaches can be captured—all using STDs.

5.4.2 ENTRY-TASK-VALIDATION-EXIT

The premise behind the development of ETVX was the necessity to find a means to embed methods and tools into a common framework for intellectual and management control. ETVX (see Figure 5-1) is a quasi-diagrammatic representation of IBM's Program Process Architecture (PPA). The authors of ETVX take the position that PPA is the highest representation of the software process and that although it contains the necessary elements for representing software engineering environments and activities, it also is applicable across a much broader framework (Radice and Phillips 1988).

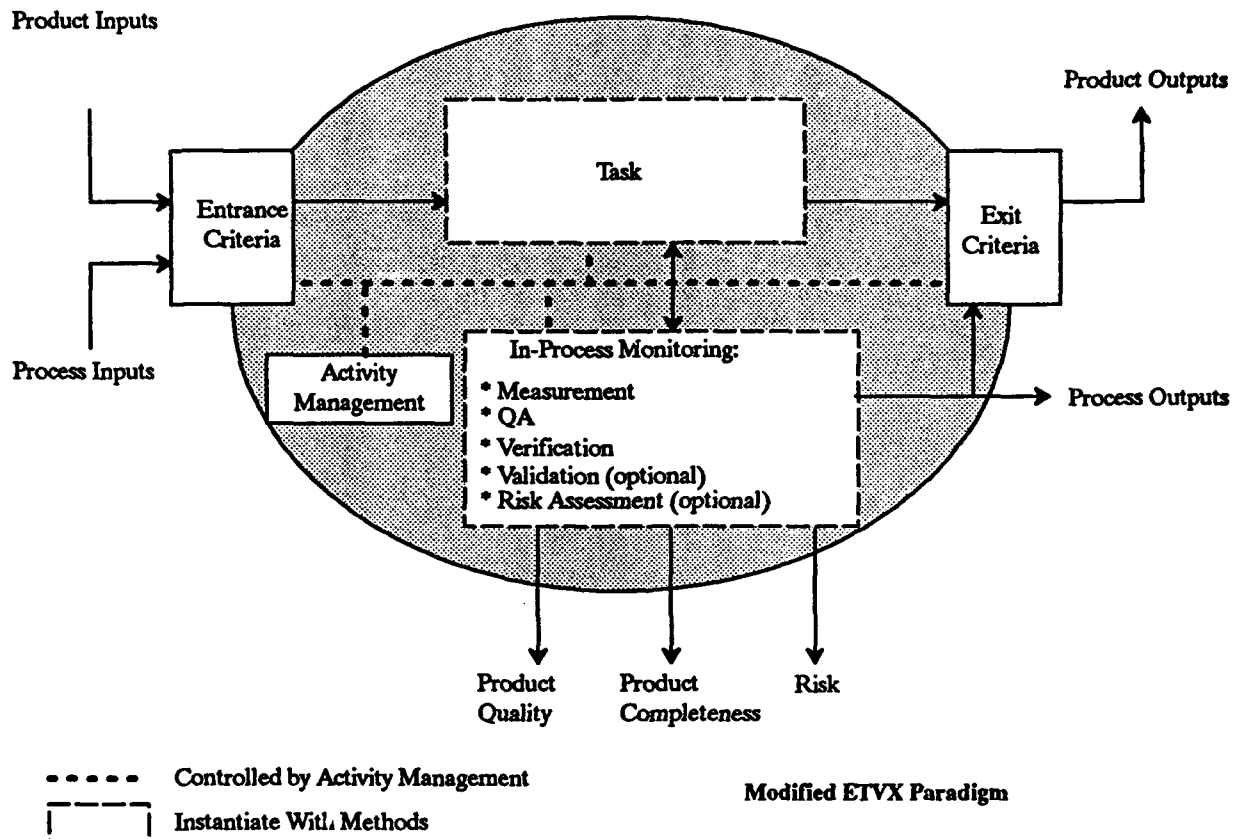


Figure 5-1. Entry-Task-Validation-eXit Diagram

Part of the motivation behind PPA was the belief that it is essential to have the ability to rigorously manage processes beyond the levels provided by individual tools and techniques. Also important was

the need to provide some means whereby an evolving process could, by virtue of how that process was enacted and applied, influence or govern the requirements for new support tools.

Generally, PPA is intended to define a basis for beginning an orderly evolution in the way that software is engineered. Radice and Phillips (1988) claim that PPA:

- Ensures a repeatable and simple paradigm at all levels of the software process
- Contains the means for self-improvement by basing itself on the need for statistical quality control
- Requires a validation mechanism for any work item produced during the development cycle
- Is based on what already exists in the software industry and draws only from the best proven alternatives
- Addresses the complete life cycle of software production
- Does not require a complete set of tools in its first iteration

The ETVX paradigm is a procedural formalism for representing activities and relationships within PPA. An ETVX box represents the concept that at any level of abstraction a work activity must have entry (E) and exit (X) criteria, some task (T) to be done, and some means or collection of means for performing validation (V).

Hierarchical decomposition is achieved in ETVX by "exploding" the task (T) component of an activity and showing the subactivities (each in ETVX form) of which it is composed. Similarly, any subactivity can also be further decomposed, as necessary. Additionally, the ETVX model does not imply all activities or tasks in succeeding stages wait for completion of preceding stages. Later stages may be functioning (that is, activities occurring) concurrent with preceding stages.

The only constraints governing the commencement of activities at a given stage are its entry criteria. Once the entry criteria are satisfied, the task related work of that stage may begin. After task related work is completed, validation may begin. Typically, the validation effort will yield information for use in evaluating the degree of compliance achieved with regard to the exit criteria. Formally, a stage is not complete until all of its exit criteria have been met. However, one or more exit criteria may be satisfied at any time during the task related work. In this way, entry criteria to other stages may become satisfied prior to complete satisfaction of exit criteria in prior stages, thereby leading to the previously mentioned parallelism and general asynchrony.

You should note that the ETVX approach does not yield an interconnected diagrammatic representation of the system being modeled. Instead, ETVX rigorously examines each of the four subcomponents that constitute an activity.

When using the templates, E and X are the Entry Criteria and Exit Criteria on the Activity template. The parent/child relations on the template can be used to define ETVX decomposition. Validation is best captured by defining a set of events that are explicitly intended for performing validation, and then heuristically asserting that all events in a given model must, somewhere within their internal processing, invoke one or more events from the validation event-tree. Internal Processing, also on the Activity template, maps to the tasks (T) in ETVX. The Child Activities, from the Process template,

explicitly describes the tasks in ETVX. Even though the templates contain all of the information for ETVX, it is important to remember that ETVX is not an operational representation. ETVX's usefulness comes from its conceptual presentation for process definition.

ETVX is useful for defining the high-level organizational processes because it has greater flexibility than the other approaches. This is especially useful in volatile environments where you may want to repeatedly change and update your representation as your understanding of the process grows. ETVX can also define a process in an environment where management is not willing to support the process control of a well defined process. ETVX is a good process representation when parts of the process, specifically some of the lower-level events, are defined in greater detail. This can provide a good compromise between high-level flexibility and low-level detail. Additionally, because ETVX has a hierarchical presentation mechanism, essential for representing large and complex processes, it represents hierarchical processes well but not as explicitly as other models.

From the perspective of the objects and relationships used within the templates, ETVX models primarily capture activity-activity, activity-product, and activity-constraint reference relations. The decomposition of activities into subactivities is also well supported. Notably absent from the ETVX approach are explicit conventions or techniques dedicated to representing the supports (roles, tools, and resources) required within a process. To capture such information, you need to use the Entry Criteria and Exit Criteria fields of ETVX, and you possibly need to add adjunct documentation to detail the characteristics of various supports.

5.4.3 STRUCTURED ANALYSIS AND DESIGN TECHNIQUE

When applying the SADT to software systems, the overall approach consists of identifying activities, identifying the inputs and outputs of those activities, identifying factors that constrain the activities, and identifying resources or materials that support the activities (Marca and McGowan 1988).

As Figure 5-2 depicts, activities are represented diagrammatically as boxes. Inputs to an activity are labeled arrows arriving at the left side of the box. Outputs from an activity are labeled arrows departing from the right side of the box. Constraining influences are labeled arrows arriving at the top of the box, and enabling mechanisms are labeled arrows arriving at the bottom of the box.

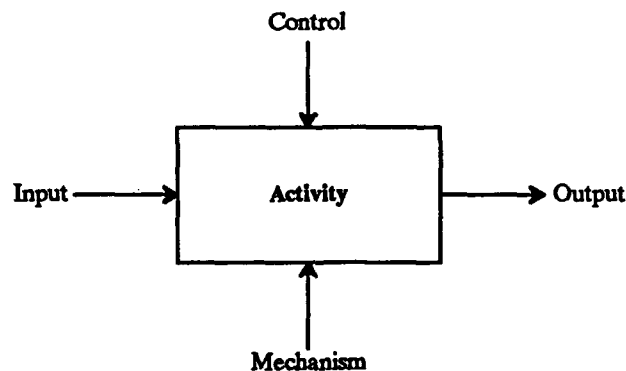


Figure 5-2. Structured Analysis and Design Technique Diagram

The outputs from one box may be the inputs, controls, or enabling mechanisms for any other box (including, in rare cases, itself). Boxes are all named and all arrows carry labels. Arrows are allowed to be split into multiple branches or join to combine multiple branches into one. Any box can be

decomposed into any number of subboxes. These, in turn, can be decomposed, and such decomposition can repeatedly continue until the necessary level of detail has been achieved. Inputs, outputs, and controls define the interfaces between boxes, and enabling mechanisms permit the controlled mixing of subjects. When a box is "exploded" to yield a new subordinate diagram, the box and diagram boundaries must match.

Figures 5-3 and 5-4 depict a small example of the top two layers of a process. Generally, and for diagrammatic clarity, a diagram is restricted to three to six boxes. This approach allows for a gradual progression in the presentation of details. Also, huge models are discouraged in favor of collections of many small, interrelated models. Each of these smaller models contributes meaning to and derives meaning from its interactions with the other small models. In principle, developing an understanding of the small models and their relationships to other small models will lead you to a clear understanding of otherwise very complex systems.

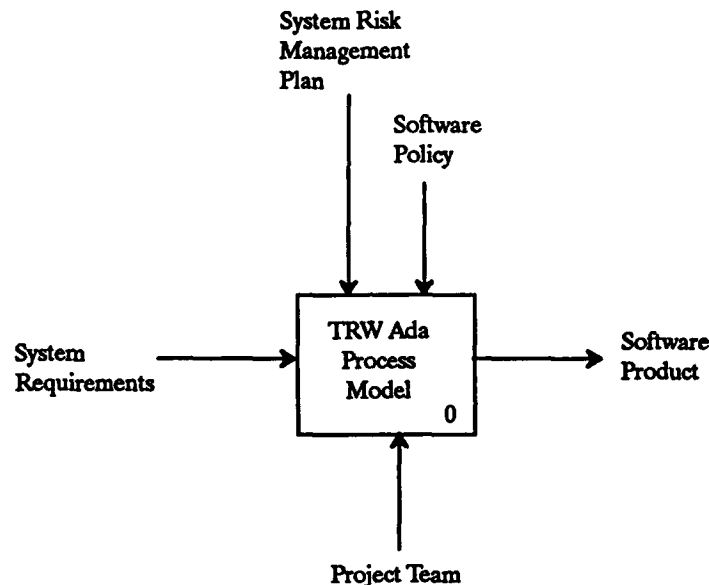


Figure 5-3. Structured Analysis and Design Technique Example 1

Commonly, it is throughputs that would be modeled in SADT as arrows arriving at the left side of a box and/or departing from the right side of a box. Constraints (especially external constraints) would be modeled as "control" arrows arriving at the top of an SADT box. The support templates (roles and resources) can be used (from the event perspective) to represent the enabling mechanisms (arrows arriving at the bottom of an SADT box). Further, SADT permits capturing a considerable amount of text information; therefore, comment and description fields and other relevant information can be readily transferred from the templates to SADT, and vice-versa.

SADT is one of the more popular software process notation being used to date. It is used for software system design, and there are a number of automated tools available on the market which support the technique. SADT is capable of large scale process definition; but SADT, like ETVX, is weak at capturing process dynamics. For example, an SADT link can only carry the syntactic structure of the process information but not the semantics of the enactment. For SADT to be enactable, you must define the type of link on the SADT diagram and then implement the semantics of the link. Along with the SADT diagram, you should also use the data dictionary for the implementation of enactment. Other limitations with SADT include the difficulty representing the concept of roles and authority for a process.

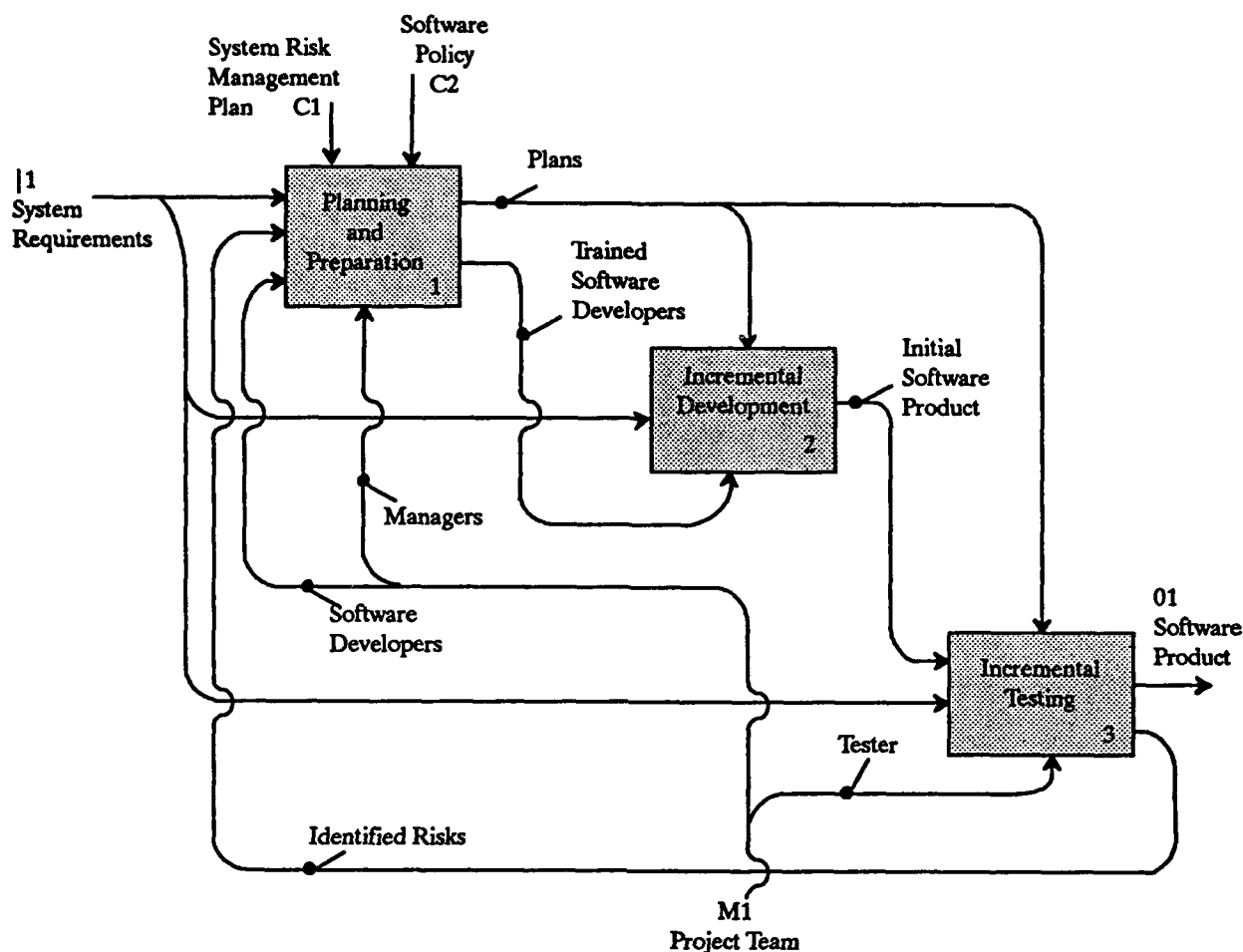


Figure 5-4. Structured Analysis and Design Technique Example 2

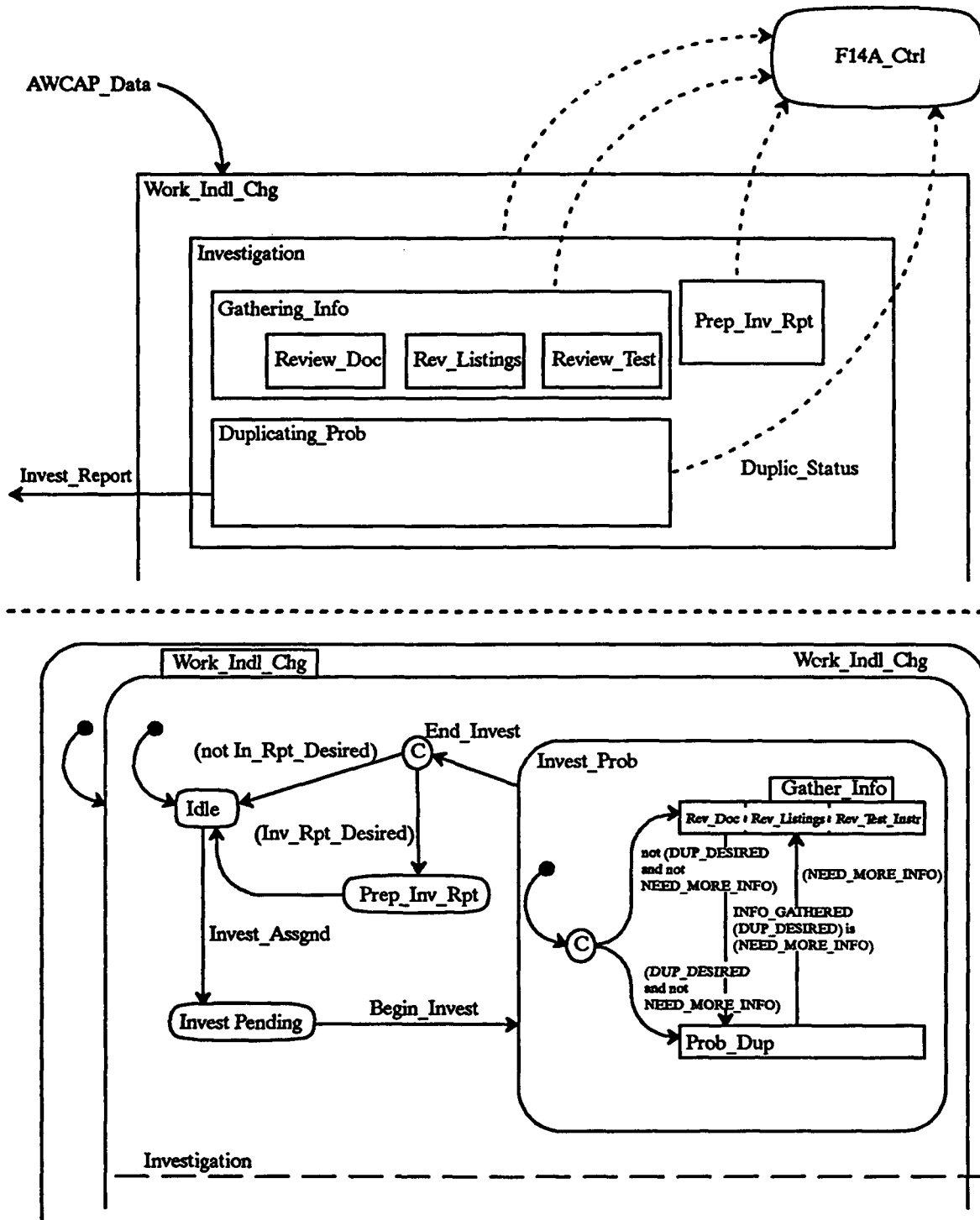
However, SADT can be relatively easy to use for defining large processes since it is capable of representing higher level abstraction and process decomposition structure.

SADT models readily capture virtually all the reference relations between and among objects. Activities can reference other activities, products, supports, and constraints. Similarly, products and supports can each reference themselves. SADT also readily supports decomposition. Although SADT primarily supports activity decomposition, deliberate use will allow SADT models to show relative decompositions of products, supports, etc. Finally, SADT does support capturing information about the sequence or order of activities, but only in the sense of constraints; this typically implies a partial-ordering, at best.

5.4.4 STATECHARTS

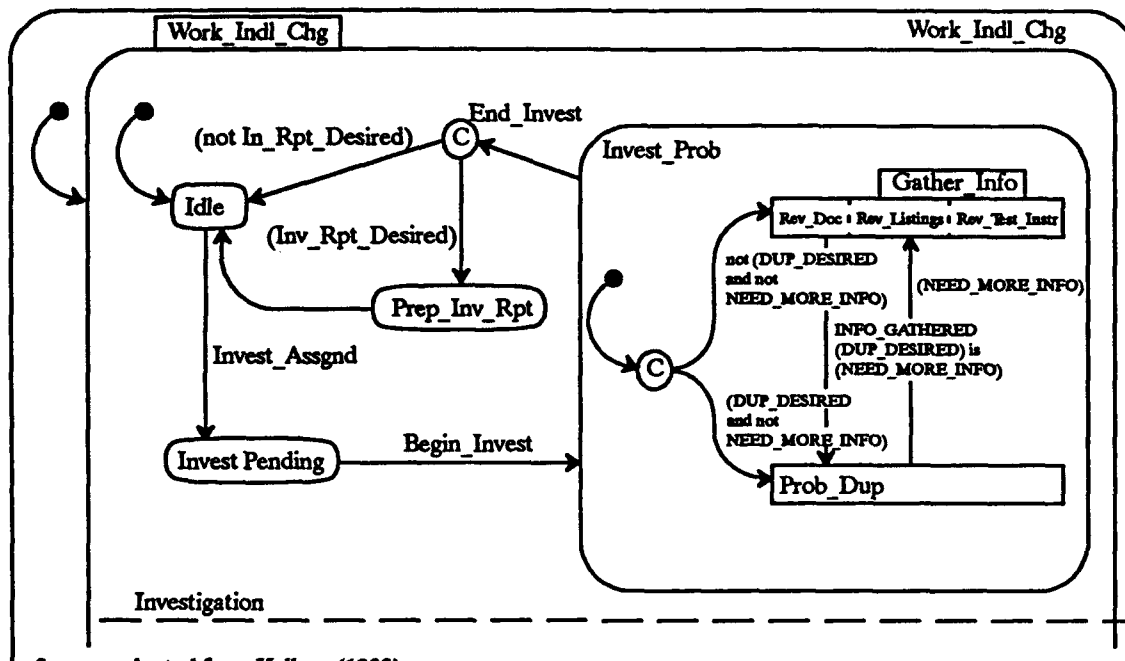
Statecharts are an extension of the basic notation used for finite state machines. Statecharts allow a finite automaton to be decomposed into a representation that models two or more interacting or communicating subsystems. Statecharts also support hierarchical decomposition of transition diagrams so that various levels of abstraction can be independently represented (Sanden 1992b). Figure 5-5 and Figure 5-6 are examples of Statecharts adapted from Marc Kellner's SEI technical report (Kellner 1989).

Figure 5-5 is an activity chart of Statestate that represents the functional perspective of process. Figure 5-6 is Statechart of Statestate that represents the behavior perspective of process.



Source: adapted from Kellner (1989)

Figure 5-5. Statechart Example 1



Source: adapted from Kellner (1989)

Figure 5-6. Statechart Example 2

Statecharts are commonly used to model concurrent, real-time system behaviors. Consequently, they are an intuitively attractive technique for general process modeling. Since statecharts were intended to model systems that are typically very complex, they contain a variety of conventions which both simplify and clarify the representations. In addition to the hierarchical decomposition already mentioned, there are techniques for representing superstates, conditional transitions, default states, history states, conditional entrance, selection-based entrance, and timeouts (Haral 1988; Coleman et al. 1990).

Hierarchical decomposition, in combination with the concept of superstates, allows gathering sets of states together that have common transitions. Typically, the superstates of a level are defined, then subordinate levels of substates are defined as an expansion of each superstate. Each such substate can, in turn, be viewed as a superstate and itself defined in terms of still lower substates. This iterative process can be repeated until the desired level of detail has been achieved.

Conditional transitions provide a simple means for extending the concept of some milestone-event causing a transition from one state to another state. Specifically, milestone events can be coupled with conditions, and then only when the right combination of milestones and conditions occurs does the state transition happen. Using this technique, it is possible to represent situations where an activity causes a state transition only if some condition is simultaneously true or where an activity results in one type of transition given one condition and the same activity results in another type of transition given a different condition. Obviously, a condition may be replaced by an arbitrarily complex conditional expression, and thereby represents increasingly complex interactions. Conditional transitions are diagrammatically represented by following an arc's listed transition (or transitions) with a "/" character which in turn is succeeded by a simple or complex conditional expression.

Default states are a simple extension within statecharts that allows a single state within each orthogonal collection of states to be labeled as its default state. Independent of whether there are one

or more orthogonal subsystems of state machines, any state machine can be marked to represent the default state of that machine. Furthermore, each subordinate refinement of superstates into substates can also include a marking that represents default states. By using this technique, it is possible to simply and explicitly represent the behavior of a machine (or "submachine") upon its receipt of initial input. Default states are represented by an unlabeled arrow pointing to one of the states within a machine.

"History states" augment the concept of "default states." With default states, the machine always commences from the indicated default, regardless of any activity which may have happened internal to that machine during a prior activation. However, history states provide a mechanism for making the entry state to a machine a function of the state that the machine was left in when last activated. As implied by the name, with history states the history of activation directly affects the behavior of future activations. (With default states, a machine's prior activity is locally unimportant—the machine always commences from the same state.) The relevance of history within a given machine is indicated by placing an unlabeled arrow pointing to a small circle surrounding the letter "H" inside the boundary rectangle of an entire machine. Note that a machine might have both a default state and history states. In such circumstances, the default state is only relevant during the first activation of that machine. All subsequent activations would be influenced by history.

Conditional entrance is an abbreviated way to represent complex transitions from one state to a number of substates. If a state, for example, transitions into one of five different substates of another state, one diagrammatic representation would be to explicitly show the five conditional transitions, each bound to the relevant substate. Alternatively, conditional entrance can simplify the diagram in the following way. Instead of showing five condition-driven variations of the event causing the transition from the first state to the second state, only one line is drawn from the first state to the second state, and it is labeled "unconditionally" as the particular event. Then the decomposition of the second state shows the conditional information as follows. The event is shown as a single inbound event on a line pointing to a small circle surrounding the letter "C" (signifying the conditional entrance). From this encircled "C" radiates (in this example) five lines, one to each of the substates. Each of these lines is then labeled with the appropriate condition or conditional expression.

Selection-based entrance is similar to conditional entrance except that selection entrance is used when decomposing a higher state into its substates and when those substates have a clearly defined one-to-one correspondence with a range of discrete values generated by the superstate. Consequently, when showing, for example, 26 different substates (representing each letter of the alphabet, any one of which might be "selected" during the superstate), it is not necessary to show 26 inbound lines, each labeled with an activity representing the selection of the appropriate letter. Instead, these 26 labeled lines can be replaced by an unlabeled arrow pointing to a small circle surrounding the letter "S." Notationally, these two approaches are equivalent, but the latter notation eliminates considerable clutter and usually results in more readily interpreted and maintained diagrams.

Finally, there are occasions, particularly with real-time system representations, where it is necessary to ensure that the system or some subsystem does not linger unnecessarily in some state. This constraint is represented in statecharts by including a small, wavy line on the left side of the upper border of the state. Under this line would be a time value indicating when this state would automatically transition to the next (and possibly prior) state.

The time constraint concept includes a lower bound in addition to the upper bound. The lower bound provides a way to represent that once a state is entered, that system (or subsystem) remains there for

at least the specified minimum time. In effect, all transition signals (or inputs), whether relevant or not, are ignored until the minimum time has passed. Neither, both, or either of these temporal constraints can be used in conjunction with a state (Haral 1988).

With these extensions, Statecharts are effectively a more robust and flexible implementation of STDs. Note that STDs can be represented using statecharts without any loss of behavioral meaning, but the converse is not true. Although relatively simple activities can often be easily diagrammed using STDs, such diagrams often become progressively more unwieldy and unmanageable as the complexity of activity increases. As a general rule, the options available in statecharts become increasingly valuable as the process being represented becomes increasingly complex.

As with STDs, the templates can be used to define a detailed state-driven model which converts almost directly into a Statechart model. Statecharts are especially good at capturing "parallel" execution of multiple state machines and can explicitly capture their interdependencies. Therefore, parallelism in events can be easily captured. However, it is important to note that if the process engineers want to eventually use Statecharts, they need to clearly note any parallelism and interdependencies that occur between steps within events.

Statecharts are good for representing critical processes where simulation and enactment are important. The tool, Statemate, will generate C or Ada code for connecting to programming libraries where various process assets can be implemented. For a given event, if the set of tasks in a process definition can be limited to a set of key fundamental activities, they can be implemented to work with code generated by Statemate. There are some potential problems with using Statecharts; they typically require low-level detail and it is difficult to represent roles and resources (supports) and research. As with many notations, Statecharts can become quite difficult to scale up for modeling highly complex, large processes. Because Statecharts capture dynamic process information, a process defined using Statecharts can be translated into a Petri net model. Dynamic models have the advantage of being subject to assessment and simulation (providing insight into time-to-completion, dead-lock, completeness, consistency, correctness, race conditions, dead-locks, and similar dynamic phenomena).

Statecharts primarily capture activities, activity sequencing and coordination, and activity decomposition. The execution of and interaction between activities is highly constrained. When properly constructed, it yields an hierarchically decomposable executable model. In practice, modeling products can be difficult because they must also be described in terms of states and state transitions. However, this can be done with the resulting model showing the relationships between changes in activity states and changes in product states: each (typically) highly constrained by the other.

5.4.5 PETRI NETS

Petri nets are progressively becoming more widely used as a means for building a wide variety of process representations. Petri nets have been successfully used to model manufacturing processes, chemical processes, hard real-time embedded processes, etc. One of the most important characteristics of Petri nets is the fact that they capture the dynamic behavioral characteristics of the system being modeled (Figure 5-7). In effect, Petri nets can be executed.

In addition to the graphical notation, Petri nets also come with a significant body of mathematical formalism. By relying on the mathematical substructure of these diagrams, it is possible to do a static structural analysis of the system's dynamic behavioral characteristics without having to resort to actually

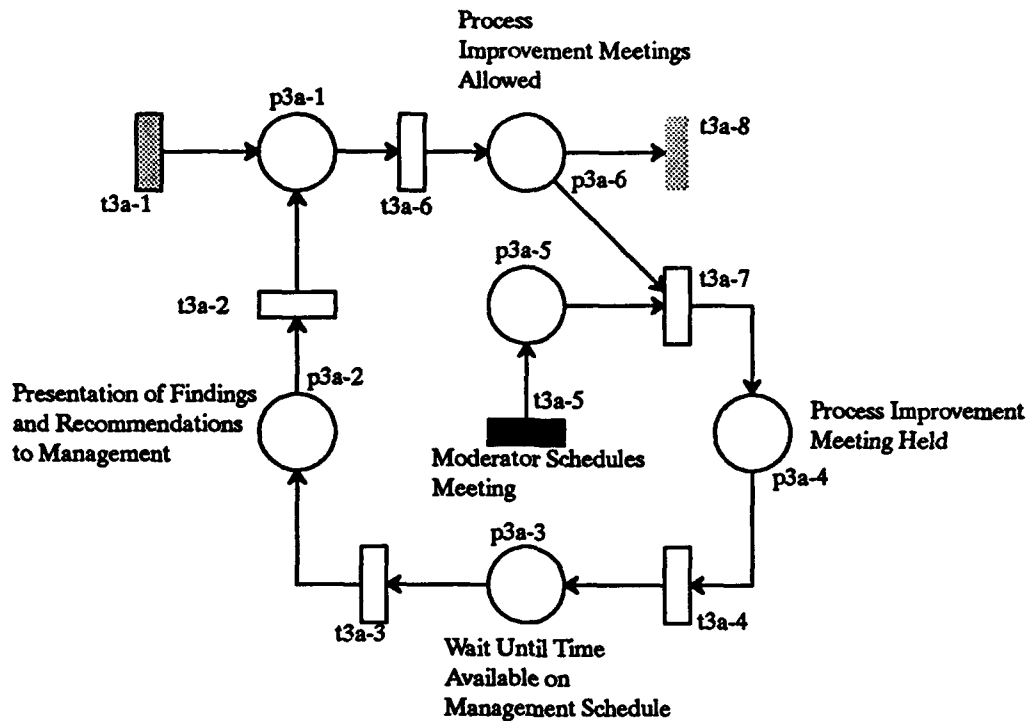


Figure 5-7. Petri-Type Net

running a simulation. This is of key importance because it allows formal interpretation and analysis of a process model for both desirable and undesirable characteristics (Levis 1992).

Another key factor contributing to the facility of Petri nets is its graphical nature. The basic graphical representation principles are conceptually simple to learn and understand, yet they can be used to build detailed representations of complex systems or models.

Formally, a Petri net is a bipartite directed multigraph that includes an initial marking. The nets are composed of two types of nodes: places and transitions that are connected by directed arcs. Arcs may connect either a place to a transition or a transition to a place. Graphically, transitions are usually depicted as bars, and places are depicted as circles.

The behavior of a Petri net is also simply described. A transition is said to be enabled if each input place to that transition carries at least as many tokens as are required by the "weight" of the connecting arc. If a transition is enabled, it may or may not fire (depending on the semantics of the model). However, if a transition does fire, the following two events occur. First, v tokens are removed from the input places (where v is the weight of the inbound connecting arc) and μ tokens are placed in each output place (where μ is the weight of the outbound connecting arc).

One simple extension to the Petri net theory includes the concept of a transition switch. When a switch is employed, only one of the output arcs fire after that transition is enabled. Consequently, instead of tokens appearing at all output places (as occurs with regular transitions) only one of the output places has a token appear. In the case of have only two output places, this behavior is like an "if, then, else" statement.

Other important extensions to basic Petri nets are the inclusion of timing and stochastic characteristics. Stochastic Petri nets are especially well suited for process modeling. These nets do not

make the assumption that transitions are instantaneous (Henderson and Taylor 1991). They include, for instance, both an enabling time, and a firing time. This introduction of time almost invariably alters the execution characteristics of these nets, just as delays in real-world activities almost invariably alter the overall schedule governing some general process.

Petri nets, in their basic form, can easily capture the relationships between events from a coordination and execution perspective, and they do so entirely from the information contained in the entry and exit conditions. More complex nets can be constructed that model availability of resources (represented by a token) and the nonviolation of constraints (represented by the absence of a token). However, the templates do not automatically request timing information, so if the eventual goal was, for example, timed stochastic Petri nets, then timing characteristics would need to be added to the information collected on the templates (and the easiest way to accommodate this is for the process engineer to modify the foundation template and add a field or two that captures timing behavior).

Petri nets are useful for process simulation and dynamic process analysis. If process execution issues are critical, Petri nets can provide what might otherwise be obscure or limited insights into dynamic behavioral characteristics (i.e., the notation itself is executable). Since Petri nets are capable of low level models, such detailed models can be more labor intensive than higher level models using other process notations. In software engineering, many of the processes being defined are still ad hoc and volatile. For some organizations, it may be best to defer constructing detailed Petri net representations, because without automated tool support such representations may be difficult to update and maintain.

The primary strength of Petri nets is their ability to capture highly constrained (and especially, temporally constrained) information about a flow of activities. Additionally, through a variety of techniques, Petri nets can be decomposed (sometimes called “unfolding”) or recomposed (“folding”). Depending on the model being constructed, subnets might be developed that represent specialized forms of high-level Petri nets. For this reason, activities and the constraints placed on those activities can be involved in specialization/generalization relationships.

5.4.6 PROCESS AND ARTIFACT STATE TRANSITION ABSTRACTION

The PASTA notation is designed to allow you to define a model for any development method or process that has defined artifacts and defined composition and dependency relations among the artifacts. Defined artifacts and relations allow for explicit artifact states upon which process states and the overall process depend. Defined artifacts allow specification of the artifacts that record software development. Artifacts and relations support analysis of product completeness. For artifacts whose completeness cannot be formally specified, you can use technical reviews and issue tracking to specify completeness.

5.4.6.1 Structure of Formal Generic Process Model

Figure 5-8 shows all of the key terms used in defining a process model. Arrows depict the relationships used in defining the design model. In the following list, each term represents one relation between two artifacts.

- **Refer-To.** Analysis on the software process always refers to an artifact, A-state, and P-state.
- **Composed-Of.** A P-state is composed of the operations that the technologist can currently perform and a role is composed of the activities that the role is able to perform.
- **Change.** An operation changes (promotes or demotes) an A-state.
- **Manipulate.** Operations manipulate artifacts.

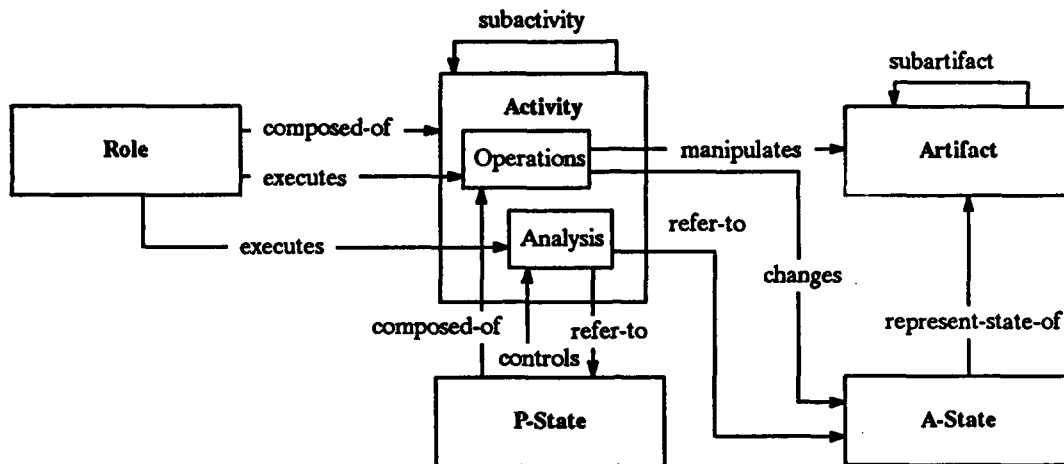


Figure 5-8. Relationships Defining the Design Model

5.4.6.2 Artifacts and Their States

Artifacts capture the decisions made during the software development process. Examples of artifacts are a description of the decomposition of a design into a set of components (such as modules, objects, packages, or subroutines) and a specification of the interface of one of the components. To characterize the state of a software development process, the engineer must characterize the state of the artifacts produced during the software process, e.g., whether or not the software decomposition is complete. However, merely characterizing the state of the artifacts is insufficient to describe a complete software process. You must also describe the activities that may be performed on artifacts, the conditions under which those activities are performed, and the roles of the people who may perform them.

5.4.6.3 Process State

The PASTA model has two levels. The lower level is based on the states of the artifacts produced during the software process; such states are called artifact states (A-states). A-states alone are insufficient to completely describe the software process, descriptions of activities, and operations on artifacts. The augmented states in the upper level state model are called process states (P-states). The two-level model allows the separation of the process description from the representation used for the artifacts.

Whether an activity is performable or not depends on the state of the artifacts. At any point in time, the set of performable activities represent the choice of artifacts on which the line engineer may work. Those that are not performable represent the artifacts on which he may not work. The model

prescribes a permissive ordering of activities (and on the work of the developer) by specifying which activities are performable and which are not performable at any point. Permissive means that the line engineer can choose from among the set of performable activities those to pursue. As with artifacts and activities, different processes will specify different roles, such as designer, reviewer, programmer, and manager. The logical grouping of activities in which people may participate defines the roles.

A P-state is represented as a rectangular box with two types, and only two types, of boxes that can intersect with it. Figure 5-9 shows an example. The two types of boxes are the entrance and exit conditions. The difference is that the entrance condition has a P-state name subbox on the bottom of the box, but the exit condition box does not. The P-state name can be on any edge of the P-state boxes. This gives the process engineer a lot of geometric and topological freedom to draw.

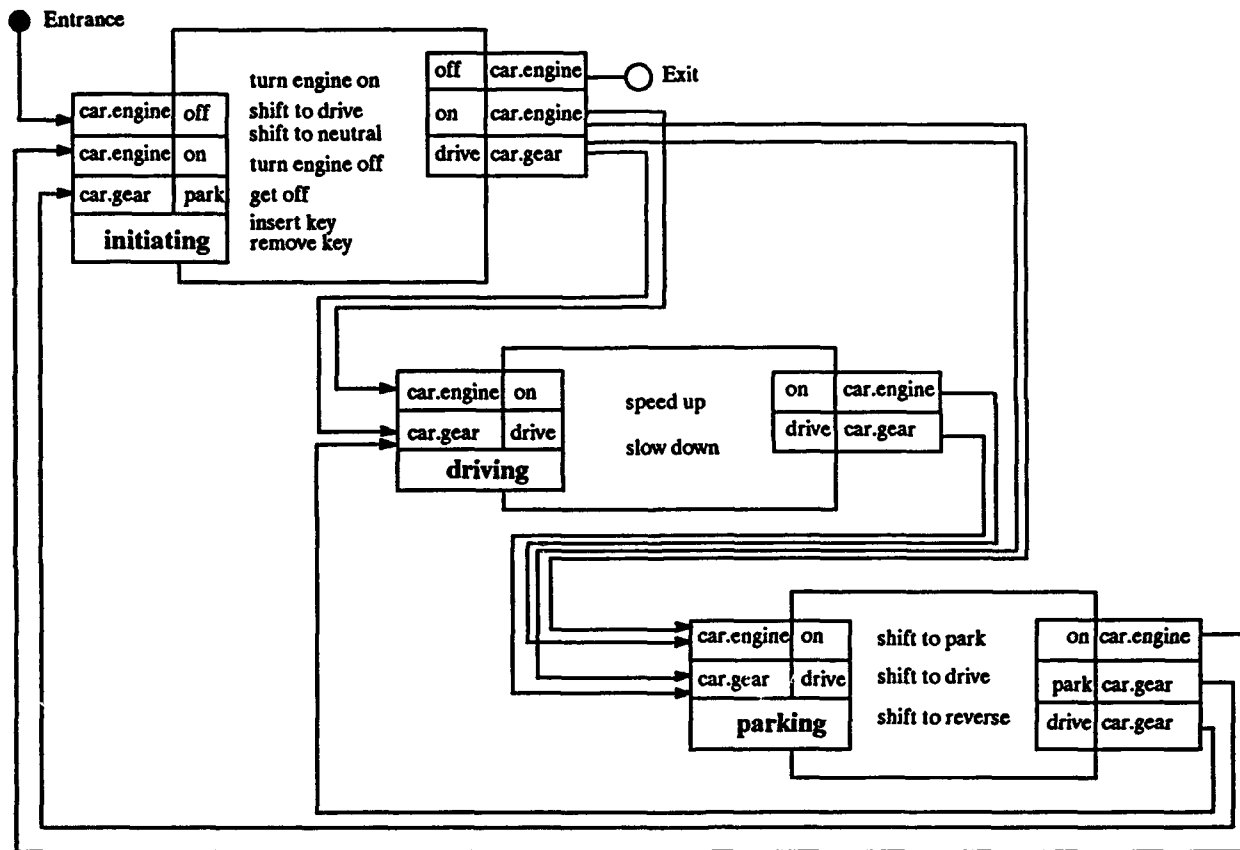


Figure 5-9. P-State Diagram Example

Figure 5-10 is an example of an artifact relation diagram. You can draw an arrow from an artifact class to an instance of an artifact. The name of the artifact has a dot in it to connect the class name and the name of the instance.

5.4.6.4 Translating From Process Templates to Process and Artifact State Transition Abstraction

The state information in the templates is certainly usable by PASTA, especially since different state-sets exist for events (processes) and throughputs (artifacts). A corporation can decide their formal migration path in different ways, e.g., from process templates to ETVX to PASTA or from the templates to PASTA directly. All of the process and activity templates can be translated to P-states. However,

if some of the activity and task templates are supported by CASE tools, these activities and tasks will be mapped to operations in PASTA. Both the template and PASTA have role elements so the mapping is direct. A role in PASTA is defined as a collection of activities. You will need to collect the set of activities for the role. All of the product, research, and resource templates will be translated to artifacts in PASTA. The cross-reference will be used extensively to translate the collection of related information which is mapped differently to PASTA. This is especially true for the difficult translation of constraints, which do not exist in PASTA. In this case, a new artifact needs to be created for supporting constraints (e.g., checklists and signatures), defined for the artifact, and then referenced in the pre- or post-conditions for either P-states or Operations. Finally, all of the super-sub relationships captured in the templates can be translated to PASTA.

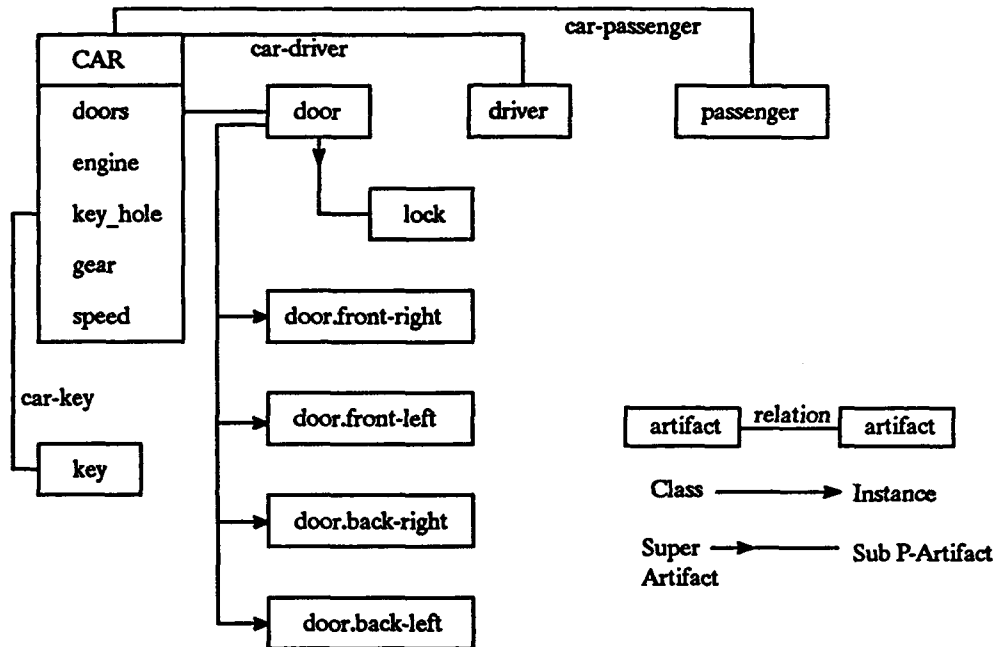


Figure 5-10. Artifact Relation Diagram Example



The primary strength of the PASTA approach is its ability to capture the relationships between activities and products in a highly constrained manner. Note that PASTA models show flow of activities as a function of state changes in underlying products. Hence, it is the state of the products that defines the state of the activity and not the converse. Again, PASTA presents what is fundamentally a two-level process model: the P-state level (process or activities) and the A-state level (artifacts or products).

5.4.7 ROLE INTERACTION NETS

A RIN is a visual formalism for the design, specification, and enactment of work processes that consist of activities ranging from informal to formal and from semiautomatic to fully automatic (Singh and Rein 1992). The RIN formalism describes processes as collections of organizational roles and their respective interactions. Role instances and their bindings to specific individuals are part of process instantiation.

A process is viewed as a collection of roles that communicate and interact to accomplish their goals. Each role in a process consists of a partially ordered set of tasks. Tasks are either solitary actions

performed without the involvement of the other roles or joint actions performed with other roles. Joint tasks can be as simple as delivering a document or as complex as negotiating a contract.

An example of the visual formalism is seen in Figure 5-11. It depicts a simple process describing the development of a document. The author, reviewer, and the repository form the set of interacting roles. Each role is represented by a column. Tasks appear as boxes. In the simple case shown here, tasks appear in the order of their execution: from top to bottom. Tasks requiring manual action are drawn as ☒. Interactions between the roles are signified by connecting their respective task with a horizontal line. The interaction between roles often involves the transfer of an artifact, such as a document or report. In the notation, a transfer is indicated by curved flow lines from the source  to its destination .

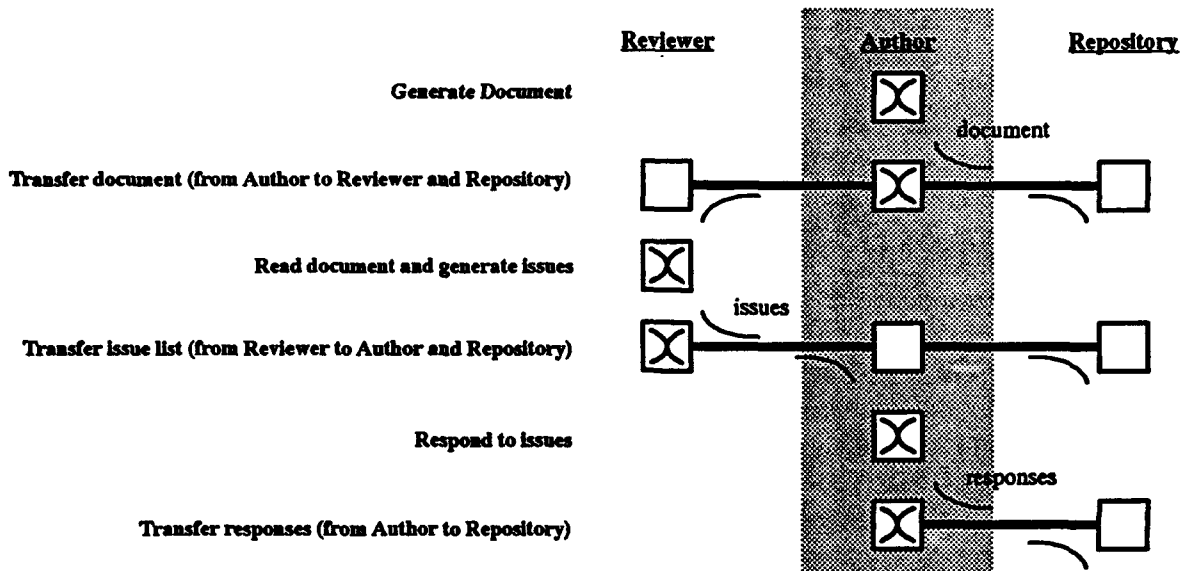


Figure 5-11. Role Interaction Net

The complete notation includes many more features than are shown in this simple example. The order of tasks within a role form a partial order and not the simple linear order implied by the example. Alternation and iteration of tasks can be specified. Further, tasks can themselves be a process, thus providing a powerful abstraction/decomposition mechanism.

Tasks have a number of attributes reminiscent of the ETVX formalism. These attributes serve as both description and additional support for process enactment. Figure 5-12 shows a task with its attributes expanded. The goal is a descriptive title for the task, e.g., Generate Document. The objects attribute lists the artifacts (inputs and outputs) necessary for performing the task. The entry and exit conditions are the necessary conditions to begin and terminate a task. In addition to these conditions, the role occupant is required to initiate and terminate all manual tasks. The execution condition defines an invariant which must be true throughout a task's execution. Failure of this condition causes the task to abort. The behavior describes the actions to be performed by the task. For automatic processes, this behavior is written in some machine-interpretable script language. Measurements define the metrics collected by the task. Finally, the permissions attribute defines the degree to which a role occupant can modify the task.

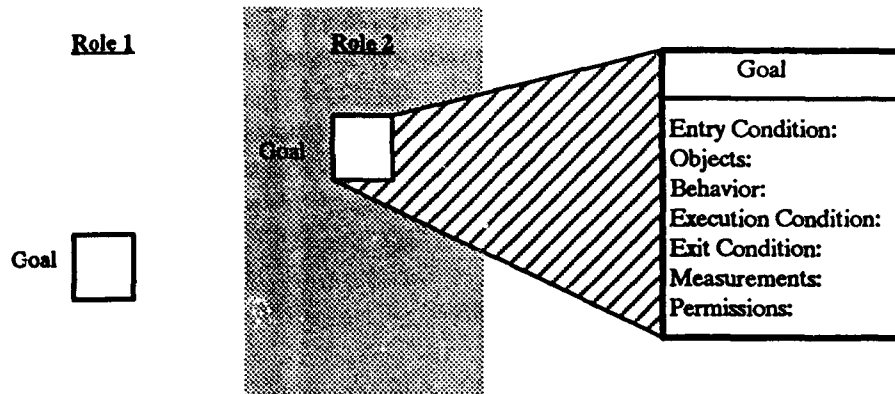


Figure 5-12. Role Interaction Net Templates

The RIN notation is supported by formally defined operational semantics. Both Singh and Rein (1992) and Singh (1992) provide formal semantics, in terms of Petri nets and STDs. These publications also provide further details and examples of the notation.

RINs seem to have a number of features that make it a desirable process modeling notation:

- Appealing visual notation
- Powerful abstraction/decomposition mechanism
- Well-defined operational semantics

However, RINs are a relatively new notation and suffer from:

- Lack of industry wide exposure
- Limited real-world application
- Few supporting tools

From the perspective of the objects and relationships defined within the templates, RINs capture four of the meta-classes: activities, products, supports, and constraints. Although inheritance is not an explicit part of the RIN model, decomposition is strongly supported through role decomposition.

5.5. BENEFITS TO THE TEMPLATE-BASED PROCESS REPRESENTATION

This section has described a small but representative selection of other notations that may be used in performing process definition and modeling. As stated previously, you can use MPDM as a preliminary effort before constructing process models based on one or more of the other process notations, or you can use the templates to extend or augment information that has already been captured using an alternative notation.

It should again be stressed that there is no approach that is universally better or universally worse for all organizations under all circumstances. Instead, the choice of a notation is highly site-specific and should involve evaluation and consideration of all the issues presented in this section of the guidebook.

If you remain uncertain on which notation to use for process definition and modeling, you should start with the templates and supporting techniques of MPDM. The primary advantage of this approach is that it is designed to allow you to start simply, confidently, and (with the use of low-cost automation) quickly allows you to automatically generate process end-products such as guidebooks and training material. Additionally, if or when the need arises, you can adapt, tailor, and even fundamentally alter the set of templates (and supporting graphical notation) to fit the specific needs of your environment. Once you are using a relatively stable set of templates, you can examine the templates and their characteristics and compare and contrast them with the characteristics of alternative notations (such as those presented in this section). The "better" notation will be, of course, the one that most completely matches your definition and modeling requirements.

Again, you can use these templates in both pre- and post-support of other process representations. More importantly, the templates are also designed to be used as a standalone technique for process definition and modeling. Sections 5.5.1 through 5.5.8 briefly present an overview of additional benefits of MPDM-based process representation by examining the templates from the perspective of the key characteristics presented in Section 5.1.

5.5.1 SCALABILITY

The templates have a very high degree of scalability. Each template contains a set of fields for representing inclusion relations. Therefore, arbitrarily long chains of ancestries can be defined. A template can be established as a parent template and other templates can serve as its "children." Any or all of those child templates may, in turn, have further (grand) children which likewise may have still other children. Consequently, you can use the templates to capture the most exacting and detailed process information, the most general or abstract process information, any level or "view" between those extremes, and any combination of these views.

5.5.2 APPLICABILITY

You will also find the templates to be highly applicable to your domain because the templates have been explicitly designed for representing organizational (as opposed to computer-executed) processes. The different template types (managerial and production activities, roles, resources, products, research, internal constraints, and external constraints) immediately provide a rich set of fundamental conventions for capturing many of the important characteristics of your process.

5.5.3 FLEXIBILITY

You will find the templates to be an extremely flexible and highly tailorable tool for process representation. The structural, hierarchical architecture that underlies the templates readily accommodates the addition of new levels to the hierarchy, new types (or "meta-classes") at the highest level, or new sublevels under any existing or new level. Additionally, you can make field and content changes to the templates at the lowest level or at higher levels where the changes can then be "inherited" by lower levels. Though the templates are completely useable as is, you can also adapt them to your changing needs.

5.5.4 READABILITY

The templates preserve the detailed readability of pure text-based representations because the majority of template fields allow unconstrained descriptive text. Additionally, the templates also

support organizational or global readability because of template interrelationships and especially because of the accompanying graphical conventions diagramming template-based process architectures and models.

5.5.5 MAINTAINABILITY

The maintainability of template-based process representations is quite high because template usage is conducive to the construction of relatively modularized process elements. Each template allows you to define an element both in terms of its internal characteristics and in terms of how that element relates to the other elements in the representation. As a result, "information hiding" (the encapsulation of implementation details so that they are invisible to the outside world) is directly achievable through template usage. You can change a template without having to perform compensating changes in numerous other templates. This yields architectures that are highly tolerant of maintenance, updates, enhancements, and similar modifications.

5.5.6 LEARNABILITY

Learnability is one of the chief benefits to the template-based approach described in this guidebook. The associated tools and techniques are comparatively quite simple, yet they can be used to capture and depict even highly complex processes. The use of these templates does not require any special background or education, and trainees can be easily constructing process representations by the end of their first day of training. There are relatively few rules that need to be followed, yet when adhered to, the result is well-defined process representations.

5.5.7 ROBUSTNESS

The templates achieve a high degree of robustness by **not** attempting to focus on just one specific domain, but instead by focusing on the phenomenon of process itself. The only assumption behind these templates is that the representation being constructed is intended to design and/or model a process. The type of process is not particularly important. Whatever that process is, if you describe the flow and relationship of activities, the passage and evolution of products and research, the people, tools, and other resources needed by that process, and the constraints and risk the process is subject to, you will capture more than enough information to understand and improve that process. You can achieve all of this by using the templates in their current form. Though tailoring the templates is certainly an option, the templates are robust enough that you may prefer—especially initially—to use them just as they are.

5.5.8 FORMALITY

The templates can support differing degrees of formality. This preserves the ability to choose the degree of formality appropriate to your needs. Since increased formality typically translates to increased cost, you may decide to use the templates to build low or moderately formal representations. These representations would be characterized by the relatively high use of free-form, text-based descriptions when providing or filling in field values. Increased formality can be achieved by, for example, using the "formal" description fields on the activity templates. Additionally, if you explicitly use states and state transitions, you can achieve a sufficiently precise and nonambiguous degree of formality that can be exported and mechanically executed within an integrated, automated process support environment.

5.5.9 FINAL REMARKS

To summarize, the benefits of template-based process representation include their versatility to be used in both top-down (functional decomposition) or bottom-up (object-oriented) methodologies for process analysis, design, development, and verification. Additionally, the template field values can be incrementally established which allows template usage to employ cyclic or spiral development techniques. There are explicit fields on the templates for referencing, or binding to, other templates so that both the characteristics of a given template and the relationships it has with other templates are quite visible and accessible. This facilitates maintaining and updating not only the information contained within the templates, but also the overall process architecture represented by the templates.

Template usage is based on a few simple conventions and requires no special background or experience; therefore, training requirements will be nominal. Furthermore, you can use them as a foundation for building representations using extensions of other notations (such as SADT or Petri nets). You can also use the templates to augment any existing representations based on other notations.

Another benefit of the template-based approach is that the use of these templates is completely consistent with the application of automated tool support. As discussed extensively in Volume 1 of this guidebook, the templates are designed as a set of "views" into a data management repository. Coupled with a "windowing" computer environment, the construction, use, and application of the templates or the resulting process representations are directly supported by a variety of commercial off-the-shelf tools or environments. Using the extensions presented in Volume 2 of this guidebook, you can achieve executable process representations that support both project-level and process-level simulation, enactment, and management.

Finally, as insurance, the templates are designed to be tailored to accommodate site-specific or organization-specific requirements. If you work under special circumstances or have uncommon requirements or needs, you can readily alter the templates so that they provide you with improved support exactly where you need it. Should your needs change, you can alter the templates to conform to the new priorities. The templates not only support you in efficiently changing your process, they also support efficiently changing the way you change your process.

APPENDIX A. INSPECTION TEMPLATES EXAMPLE

Section 5 of Volume 1 includes an example of templates used to define an Inspection Process. Only a subset of the templates were shown. This appendix contains the full set of example templates. Section A.1 activity templates, Section A.2 shows product templates, and Section A.3 example role templates.

In this appendix, an ellipses in a section title indicates a group heading as opposed to an actual field.

A.1 PEER REVIEW ACTIVITY TEMPLATES

A.1.1 ACTIVITY TEMPLATE: PEER REVIEWS

A.1.1.1 Name

Peer Reviews

A.1.1.2 Unique ID

INS

A.1.1.3 Brief Description

The peer review activity is based upon the formal inspection process.

A.1.1.4 Overview Description

The inspection method is one technique for the static verification of an artifact. Developed by Michael E. Fagan of IBM (Fagan 1984), the primary objective of this method is to detect defects in an artifact in an efficient and effective manner. Frequently referred to as a "formal inspection," you can use this method to verify any artifact generated during the development process, although considerable focus has been given to applying this method to preliminary design, detailed design, and code artifacts.

The inspection method consists of well-defined roles and activity stages. Associated with the roles are specific responsibilities. Additionally, entrance and exit criteria and objectives for each activity stage and mechanisms for recording and reporting defects ensure consistent application of this method.

The efficiency of this approach depends on focusing the participants on the overall objective of the method, which is defect detection. The effectiveness of this method depends on proper training of the participants and commitment by management to adhere to the prescribed process.

As stated earlier, you can use the inspection method to verify any artifact generated during the development process. This section, however, states the specific purpose for preliminary design, detailed design, and code inspections.

During the preliminary design phase, you hold preliminary design inspections to inspect the design documentation to verify that the requirements are satisfied and that the product will be maintainable, adaptable, and of high quality.

During the detailed design phase, you hold detailed design inspections to inspect the design documentation to verify that the design has been refined correctly, is defined to a level which allows coding to begin, and satisfies assigned product requirements. You inspect the design to ensure that it is maintainable, adaptable, and of high quality.

During the implementation phase, you hold code inspections to inspect the code to verify that it implements the detailed design correctly and completely.

The project team must baseline all reference material (i.e., specifications, requirements, documents) prior to the inspection meeting. For example, a preliminary design inspection cannot be held until they

have baselined the requirements. An example of a requirements specification document is a system requirements specification (SRS).

A.1.1.5 Summary Description

It should also be noted that this method supports two types of tailoring. The first includes tailoring of guidelines and checklists to create an inspection process which adheres to the method and is relevant to the organization. You must do this type of tailoring to ensure a valid inspection process. The second type of tailoring is to modify the inspection method. This type of tailoring is optional. Additionally, tailoring of the method is subject to the limitations described below. The following discussion presents the mandatory tailoring followed by the optional tailoring.

Use of the inspection method requires that your organization establishes supporting mechanisms. These supporting mechanisms include entry criteria checklists, preparation checklists, and reinspection criteria for each artifact type that will be inspected. While there are many examples of these, you must tailor them to your organization's goals and the specific artifact being inspected.

Your organization may also tailor the classification of the defects. Typically, you classify defects, at a minimum, by type and severity. You may also classify each defect by category (i.e., missing, wrong, extra).

Because the objective of the inspection method is to find defects early in an efficient and effective manner, the focus is clearly on **defect detection**. One approach to tailoring the method, therefore, is to remove the causal analysis stage. The results of the causal analysis activity are intended to improve the development process to prevent the defect from occurring in the future. This stage emphasizes **defect prevention**. For organizations introducing the inspection process, the initial focus is probably on defect detection. The causal analysis activity could then be added in the future.

A second type of optional tailoring is to combine multiple inspections into a single inspection. This situation seems more frequent when inspecting a software artifact. An example is a change that is small in scale. The author may want to consider combining the preliminary design and detailed design inspections. Guidelines for determining when multiple inspections could be combined are:

- Combine the preliminary design and detailed design inspections when an update is made to an existing module and the preliminary design is not affected.
- Combine the detailed design and code inspections when the detailed design is done at a very detailed level and is expressed in the target language.

The moderator should be certified to run an inspection. Certification should consist of a minimum of 1 day of training in the inspection process and the specific role of the moderator within that process. In addition, newly trained moderators should be observed by an experienced moderator for the first two to three inspections. Other participants in an inspection should be given a 1-hour overview of the inspection process.

A.1.1.6 Parent Template(s)

<Field is Empty>

A.1.1.7 Child Templates...

A.1.1.7.1 Unique ID

INS_PLAN

A.1.1.7.1.1 Child Type

Composition

A.1.1.7.1.2 Child Tailoring

Required

A.1.1.7.1.3 Child Proximity

Local

A.1.1.7.2 Unique ID

INS_OVR

A.1.1.7.2.1 Child Type

Composition

A.1.1.7.2.2 Child Tailoring

Suggested

A.1.1.7.2.3 Child Proximity

Local

A.1.1.7.3 Unique ID

INS_PREP

A.1.1.7.3.1 Child Type

Composition

A.1.1.7.3.2 Child Tailoring

Recommended

A.1.1.7.3.3 Child Proximity

Local

A.1.1.7.4 Unique ID

INS_MTG

A.1.1.7.4.1 Child Type

Composition

A.1.1.7.4.2 Child Tailoring

Recommended

A.1.1.7.4.3 Child Proximity

Local

A.1.1.7.5 Unique ID

INS_REWK

A.1.1.7.5.1 Child Type

Composition

A.1.1.7.5.2 Child Tailoring

Suggested

A.1.1.7.5.3 Child Proximity

Local

A.1.1.7.6 Unique ID

INS_FWUP

A.1.1.7.6.1 Child Type

Composition

A.1.1.7.6.2 Child Tailoring

Suggested

A.1.1.7.6.3 Child Proximity

Local

A.1.1.8 Used Within...

A.1.1.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.1.8.1.1 Output Object Type

Procedure_Manual

A.1.1.8.1.2 Include In Next Build

Yes

A.1.1.9 Transition Text...

A.1.1.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.1.9.1.1 Leading Transition Text

The technical office has decided that the Peer Review process used internally will be a variant of the Fagan formal inspection method. Comments and recommendations for improving either this material in particular, or the peer review process in general, should be forwarded through your local SEPG representative.

A.1.1.9.1.2 Trailing Transition Text

<Field is Empty>

A.1.1.10 Activity Criteria and Process...

A.1.1.10.1 Entry Criteria

The author or project manager has selected a trained moderator for the preliminary design, detailed design, or code inspection.

The moderator has agreed to moderate the indicated inspection.

The moderator has been notified by the author that the preliminary design, detailed design, or code is completed, baselined, and is ready for examination.

A.1.1.10.2 Internal Process

There are seven parts to the inspection process. Six of these parts are required. These include: planning, preparation, inspection, causal analysis, rework, and follow-up. The optional part is the overview. The following sections indicate the entrance criteria, procedure, and exit criteria for each of these parts.

A.1.1.10.3 Exit Criteria

The Inspection Close Memo has been completed and distributed.

A.1.1.10.4 Invariants

The product being inspected continues to have purpose.

A.1.1.11 Related Products...**A.1.1.11.1 Unique ID**

IIP

A.1.1.11.1.1 Usage

Before Starting

A.1.1.11.1.2 Impact

Left Unchanged

A.1.1.11.1.3 Tailoring Options

Required

A.1.1.11.1.4 Elaboration Text

<Field is Empty>

A.1.1.11.2 Unique ID

MEMO

A.1.1.11.2.1 Usage

While Underway

A.1.1.11.2.2 Impact

Created

A.1.1.11.2.3 Tailoring Options

Recommended

A.1.1.11.2.4 Elaboration Text

<Field is Empty>

A.1.1.11.3 Unique ID

RMET

A.1.1.11.3.1 Usage

Before Ending

A.1.1.11.3.2 Impact

Created

A.1.1.11.3.3 Tailoring Options

Required

A.1.1.11.3.4 Elaboration Text

<Field is Empty>

A.1.1.12 Related Supports...

A.1.1.12.1 Unique ID

PRT

A.1.1.12.1.1 Usage

While Underway

A.1.1.12.1.2 Tailoring Options

Required

A.1.1.12.1.3 Elaboration Text

<Field is Empty>

A.1.1.12.2 Unique ID

RCO

A.1.1.12.2.1 Usage

Before Starting

A.1.1.12.2.2 Tailoring Options

Required

A.1.1.12.2.3 Elaboration Text

<Field is Empty>

A.1.2 ACTIVITY TEMPLATE: INSPECTION PLANNING

A.1.2.1 Name

Inspection Planning

A.1.2.2 Unique ID

INS_PLAN

A.1.2.3 Brief Description

The Inspection Planning activity plans the time and resources of the inspection process.

A.1.2.4 Overview Description

During the planning stage, the author (i.e., owner) gives a copy of the material to be inspected to the moderator, who verifies the inspection readiness of the artifact. The moderator typically completes this assessment by using an entry criteria checklist suitable for the artifact type being inspected. If the artifact is ready to be inspected, the moderator and the author of the artifact identify relevant reference material, a checklist of likely defects (i.e., a preparation checklist), the participants and their respective roles, and the inspection schedule. The moderator generates the preparation checklist, which is suitable for the artifact type being inspected, by modifying a generic checklist.

The moderator uses planning guidelines, suitable for the artifact type being inspected, to aid in determining the inspection schedule. These guidelines include limiting the length of a single inspection meeting to 2 hours and not scheduling more than 4 hours per day to inspection activities. The moderator can derive guidelines to determine the duration of the overview, preparation, and inspection meeting activities from the following:

A.1.2.5 Summary Description

<Field is Empty>

A.1.2.6 Parent Template(s)

INS

A.1.2.7 Child Templates...

<Fields are Empty>

A.1.2.8 Used Within...

A.1.2.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.2.8.1.1 Output Object Type

Procedure_Manual

A.1.2.8.1.2 Include In Next Build

Yes

A.1.2.9 Transition Text...

<Fields are Empty>

A.1.2.10 Activity Criteria and Process...

A.1.2.10.1 Entry Criteria

The author submits a completed entry criteria checklist with the inspection package to the moderator. Refer to the appropriate procedure for the details of the entrance criteria checklists.

A.1.2.10.2 Internal Process

The moderator examines the materials to determine:

- The correct level of detail
- Whether it is an appropriate amount to inspect
- The availability of appropriate reference documents

The moderator calculates the time required for the overview meeting, preparation time, and inspection meeting by dividing the size of the document to be inspected, in pages, or the code to be inspected, in lines of code, by the following estimates (these estimates are based on data related to an inspection process used in another organization and will be revised as data is collected):

- **Overview.** An overview should be limited to a single two-hour meeting. If an overview longer than 2 hours is indicated by the length of the material, the moderator and the author should examine the material to divide it into separate inspection pieces. If the material indicates that an overview can be completed in a half-hour or less, the overview can be held at the beginning of the inspection meeting.
- **Preparation.** An estimate for preparation rates for preliminary design material is 12 to 15 pages per hour; for detailed design material, 8 to 12 pages per hour; and for code, 200 noncomment source statements per hour.
- **Inspection.** An estimate for inspection rates for preliminary design material is 10 to 12 pages per hour; for detailed design material, 8 to 10 pages per hour; and for code, 100 to 150 noncomment source statements per hour.

The moderator, project manager, or author contacts the inspection participants, tells them what is expected, and solicits available dates.

If an overview meeting is desired, it is scheduled by the moderator.

The moderator schedules the inspection using the estimated inspection meeting time computed earlier. An inspection may require multiple meetings. The length of a single inspection meeting should not exceed 4 hours in a given day. For each 4-hour meeting, it should be organized into a 2-hour meeting, a half-hour break, and then the remaining 2 hours.

The moderator or author (with approval of the moderator) distributes the inspection package consisting of the reference and inspection materials to the participants after appending the following:

- The Inspection Invitation Memo which identifies the type of inspection, time, location, time charge number, participants, and job assignments for the inspection
- Trivial Error Log

A.1.2.10.3 Exit Criteria

If an overview meeting is indicated, the overview meeting has been scheduled.

The inspection meeting has been scheduled.

The inspection package has been distributed to all inspection participants.

A.1.2.10.4 Invariants

<Field is Empty>

A.1.2.11 Related Products...

A.1.2.11.1 Unique ID

RMET

A.1.2.11.1.1 Usage

Before Starting

A.1.2.11.1.2 Impact

Left Unchanged

A.1.2.11.1.3 Tailoring Options

Required

A.1.2.11.1.4 Elaboration Text

<Field is Empty>

A.1.2.11.2 Unique ID

MEMO_INV

A.1.2.11.2.1 Usage

Before Ending

A.1.2.11.2.2 Impact

Created

A.1.2.11.2.3 Tailoring Options

Required

A.1.2.11.2.4 Elaboration Text

<Field is Empty>

A.1.2.12 Related Supports...

A.1.2.12.1 Unique ID

RCO

A.1.2.12.1.1 Usage

Before Starting

A.1.2.12.1.2 Tailoring Options

Required

A.1.2.12.1.3 Elaboration Text

<Field is Empty>

A.1.2.12.2 Unique ID

PRT_MOD

A.1.2.12.2.1 Usage

Before Starting

A.1.2.12.2.2 Tailoring Options

Required

A.1.2.12.2.3 Elaboration Text

<Field is Empty>

A.1.2.12.3 Unique ID

PRT_DEV

A.1.2.12.3.1 Usage

Before Starting

A.1.2.12.3.2 Tailoring Options

Suggested

A.1.2.12.3.3 Elaboration Text

<Field is Empty>

A.1.3 ACTIVITY TEMPLATE: INSPECTION OVERVIEW

A.1.3.1 Name

Inspection Overview

A.1.3.2 Unique ID

INS_OVR

A.1.3.3 Brief Description

The Inspection Overview activity provides the context of the Inspection activity and explanation of the product to be inspected.

A.1.3.4 Overview Description

The overview consists of a presentation, by the author, of a general description of the inspection material to the remaining participants. Additionally, the moderator may use the overview meeting to emphasize the purpose of the inspection method and to ensure that the participants understand their roles and responsibilities. This meeting is typically held prior to the inspection meeting.

A.1.3.5 Summary Description

<Field is Empty>

A.1.3.6 Parent Template(s)

INS

A.1.3.7 Child Templates...

<Fields are Empty>

A.1.3.8 Used Within...

A.1.3.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.3.8.1.1 Output Object Type

Procedure_Manual

A.1.3.8.1.2 Include In Next Build

Yes

A.1.3.9 Transition Text...

<Fields are Empty>

A.1.3.10 Activity Criteria and Process...**A.1.3.10.1 Entry Criteria**

The planning stage has been completed.

All key inspectors are present.

A.1.3.10.2 Internal Process

The moderator runs the meeting.

The author presents his design, addressing the following:

- The purpose of the functional changes/additions
- The external interfaces
- The relation between the modules, as a function, to the rest of the software component and to the software product

A.1.3.10.3 Exit Criteria

The overview has been presented.

A.1.3.10.4 Invariants

<Field is Empty>

A.1.3.11 Related Products...**A.1.3.11.1 Unique ID**

IIP

A.1.3.11.1.1 Usage

Before Starting

A.1.3.11.1.2 Impact

Left Unchanged

A.1.3.11.1.3 Tailoring Options

Required

A.13.11.1.4 Elaboration Text

<Field is Empty>

A.13.12 Related Supports...

A.13.12.1 Unique ID

PRT

A.13.12.1.1 Usage

Before Starting

A.13.12.1.2 Tailoring Options

Required

A.13.12.1.3 Elaboration Text

<Field is Empty>

A.1.4 ACTIVITY TEMPLATE: INSPECTION PREPARATION

A.1.4.1 Name

Inspection Preparation

A.1.4.2 Unique ID

INS_PREP

A.1.4.3 Brief Description

The Inspection Preparation activity consists of each inspector individually reviewing the product.

A.1.4.4 Overview Description

During the preparation stage, each inspector individually reviews the inspection material. During this individual review, he verifies the artifact for technical accuracy, fulfillment of requirements, and adherence to standards and conventions. The inspector records any defects discovered during this activity stage. He may also develop questions which may lead to the discovery of additional defects during the inspection meeting. The moderator ensures that adequate time has been allocated for proper preparation and assists the inspectors, as needed, in their preparation activity.

A.1.4.5 Summary Description

<Field is Empty>

A.1.4.6 Parent Template(s)

INS

A.1.4.7 Child Templates...

<Fields are Empty>

A.1.4.8 Used Within...

A.1.4.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.4.8.1.1 Output Object Type

Procedure_Manual

A.1.4.8.1.2 Include In Next Build

Yes

A.1.4.9 Transition Text...

<Fields are Empty>

A.1.4.10 Activity Criteria and Process...

A.1.4.10.1 Entry Criteria

The planning stage has been completed if a separate overview is not planned. Otherwise, the overview has been completed.

A.1.4.10.2 Internal Process

The moderator ensures that each inspector has received the inspection package. This may be done by phone, electronic mail, or personal visit.

The moderator ensures that sufficient preparation time is available for each inspector.

Each participant prepares for the inspection. During preparation, trivial errors are documented on the Trivial Error Log. Refer to the appropriate procedure for guidelines for reviewing the inspection material.

A.1.4.10.3 Exit Criteria

Each inspector has the materials, knows his role, has had sufficient time to prepare, knows the place and time of the inspection, and has a copy of the Trivial Error Log.

The moderator determines that all key inspectors are prepared for the inspection.

A.1.4.10.4 Invariants

<Field is Empty>

A.1.4.11 Related Products...

A.1.4.11.1 Unique ID

IIP

A.1.4.11.1.1 Usage

Before Starting

A.1.4.11.1.2 Impact

Left Unchanged

A.1.4.11.1.3 Tailoring Options

Required

A.1.4.11.1.4 Elaboration Text

<Field is Empty>

A.1.4.12 Related Supports...

A.1.4.12.1 Unique ID

PRT

A.1.4.12.1.1 Usage

Before Starting

A.1.4.12.1.2 Tailoring Options

Required

A.1.4.12.1.3 Elaboration Text

<Field is Empty>

A.1.5 ACTIVITY TEMPLATE: INSPECTION MEETING

A.1.5.1 Name

Inspection Meeting

A.1.5.2 Unique ID

INS_MTG

A.1.5.3 Brief Description

<Field is Empty>

A.1.5.4 Overview Description

At the inspection meeting, the inspectors report defects and the moderator records a description of the defect. The inspector who identifies a defect additionally classifies that defect according to its type and its severity. The moderator pursues questions that are raised only to the point where the answers are found to be satisfactory or a defect is identified and recorded. **Fixing defects is not the purpose of the meeting.**

At the conclusion of this meeting, the moderator completes a brief "process check." The purpose of this activity is to give each participant the opportunity to critique the Inspection Meeting activity. The moderator uses results of this process check to improve the effectiveness and/or efficiency of the inspection meeting.

A.1.5.5 Summary Description

<Field is Empty>

A.1.5.6 Parent Template(s)

INS

A.1.5.7 Child Templates...

A.1.5.7.1 Unique ID

INS_MTG_REV

A.1.5.7.1.1 Child Type

Composition

A.1.5.7.1.2 Child Tailoring

Required

A.1.5.7.1.3 Child Proximity

Local

A.1.5.7.2 Unique ID

INS_MTG_REIN

A.1.5.7.2.1 Child Type

Composition

A.1.5.7.2.2 Child Tailoring

Required

A.1.5.7.2.3 Child Proximity

Local

A.1.5.8 Used Within...

A.1.5.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.5.8.1.1 Output Object Type

Procedure_Manual

A.1.5.8.1.2 Include In Next Build

Yes

A.1.5.9 Transition Text...

<Fields are Empty>

A.1.5.10 Activity Criteria and Process...

A.1.5.10.1 Entry Criteria

The preparation stage has been completed.

A.1.5.10.2 Internal Process

The moderator runs the inspection and:

- Ensures that the inspection begins on time.
- Introduces people and identifies their role in the inspection.
- Solicits and records the preparation time of each inspector. All inspectors should submit their preparation time on the Trivial Error Log to obtain an accurate accounting of the preparation time.
- Reschedules the inspection if the moderator determines, at the start of the inspection, that the preparation by any of the key inspectors was inadequate or that a key inspector is absent. In this case, the project manager is to be notified.

The reader pages through the inspection material, paraphrasing as appropriate.

The inspectors identify errors as the reader steps through the material.

The moderator records the errors on the Inspection Error Log and categorizes them, soliciting input from the inspection participants.

The moderator summarizes the error and the categorization.

At the end of the inspection meeting, all participants make a reinspection determination. A reinspection is required if a simple majority feels that it is necessary.

A.1.5.10.3 Exit Criteria

The inspection meeting has been held.

All inspection material (design or code) has been inspected.

All detected errors have been noted on the Inspection Error Log.

The reinspection decision has been made.

A.1.5.10.4 Invariants

<Field is Empty>

A.1.5.11 Related Products...

A.1.5.11.1 Unique ID

IIP

A.1.5.11.1.1 Usage

Before Starting

A.1.5.11.1.2 Impact

Left Unchanged

A.1.5.11.1.3 Tailoring Options

Required

A.1.5.11.1.4 Elaboration Text

<Field is Empty>

A.1.5.11.2 Unique ID

MEMO_RES

A.1.5.11.2.1 Usage

Before Starting

A.1.5.11.2.2 Impact

Left Unchanged

A.1.5.11.2.3 Tailoring Options

Required

A.1.5.11.2.4 Elaboration Text

<Field is Empty>

A.1.5.12 Related Supports...

A.1.5.12.1 Unique ID

PRT

A.1.5.12.1.1 Usage

Before Starting

A.1.5.12.1.2 Tailoring Options

Required

A.1.5.12.1.3 Elaboration Text

<Field is Empty>

A.1.6 ACTIVITY TEMPLATE: INSPECTION MEETING REVIEW

A.1.6.1 Name

Inspection Meeting Review

A.1.6.2 Unique ID

INS_MTG_REV

A.1.6.3 Brief Description

<Field is Empty>

A.1.6.4 Overview Description

The errors discovered at an inspection meeting are recorded and categorized along two dimensions, with the exception that trivial errors are classified along one dimension.

The first dimension identifies the type of error. One example of this categorization for software artifacts includes requirements, design, data, logic, syntax, standards, interface, return code/message/exception, prologue/comment, and performance. A brief description of each of these is as follows:

- **Requirements**. The error is attributed to a requirement that is either unclear, incorrect, or inconsistent.
- **Design**. The error is attributed to the design. Typically, the design does not meet the baselined requirements specification.
- **Data**. The error reflects a missing, extra, or wrong data definition. It can also indicate incorrect use of a data definition.
- **Logic**. The error is algorithmic in nature and can refer to missing, wrong, or extra logic.
- **Syntax**. The error violates the grammar of the defined design or code language.
- **Standards**. The error reflects a deviation from organization or project standards.
- **Interface**. The error refers to an incompatible definition or format of the information being exchanged between two components.
- **Return Code/Message/Exception**. The error indicates that the wrong return code/message/exception is being returned to the calling routine.
- **Prologue/Comment**. The explanation accompanying the design or code is incorrect, implicit, or missing.
- **Performance**. The code will not perform in the amount of time or resources allocated if implemented as designed/coded.

The second dimension identifies the severity of the error and consists of two primary categories, major and minor, and one secondary category, trivial. The distinction between these is as follows:

- **Major.** A major error is a deviation between the intended and observed state or behavior of a system.
- **Minor.** A minor error indicates a violation of standards, conventions, or rules which would not result in a deviation from requirements if not corrected, but could result in difficulties in terms of maintenance or clarity of purpose.
- **Trivial.** A trivial error is a formatting, typographical, or spelling defect whose correct interpretation is obvious to a knowledgeable reader.

The inspector shall make the judgment in categorizing errors along these two dimensions. The moderator shall solicit this input from the inspection participants at the time the error is being recorded.

A.1.6.5 Summary Description

<Field is Empty>

A.1.6.6 Parent Template(s)

INS_MTG

A.1.6.7 Child Templates...

<Fields are Empty>

A.1.6.8 Used Within...

A.1.6.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.6.8.1.1 Output Object Type

Procedure_Manual

A.1.6.8.1.2 Include In Next Build

Yes

A.1.6.9 Transition Text...

<Fields are Empty>

A.1.6.10 Activity Criteria and Process...

A.1.6.10.1 Entry Criteria

A.1.6.10.2 Internal Process

A.1.6.10.3 Exit Criteria

A.1.6.10.4 Invariants

<Field is Empty>

A.1.6.11 Related Products...

<Fields are Empty>

A.1.6.12 Related Supports...

<Fields are Empty>

A.1.7 ACTIVITY TEMPLATE: INSPECTION MEETING REINSPECTION DECISION**A.1.7.1 Name**

Inspection Meeting Reinspection Decision

A.1.7.2 Unique ID

INS_MTG_DEC

A.1.7.3 Brief Description

<Field is Empty>

A.1.7.4 Overview Description**A.1.7.5 Summary Description**

<Field is Empty>

A.1.7.6 Parent Template(s)

INS_MTG

A.1.7.7 Child Templates...

<Fields are Empty>

A.1.7.8 Used Within...**A.1.7.8.1 Output Object Name**

Peer_Review_Procedure_Manual

A.1.7.8.1.1 Output Object Type

Procedure_Manual

A.1.7.8.1.2 Include In Next Build

Yes

A.1.7.9 Transition Text...

<Fields are Empty>

A.1.7.10 Activity Criteria and Process...**A.1.7.10.1 Entry Criteria****A.1.7.10.2 Internal Process****A.1.7.10.3 Exit Criteria****A.1.7.10.4 Invariants**

<Field is Empty>

A.1.7.11 Related Products...

<Fields are Empty>

A.1.7.12 Related Supports...

<Fields are Empty>

A.1.8 ACTIVITY TEMPLATE: INSPECTION REWORK**A.1.8.1 Name**

Inspection Rework

A.1.8.2 Unique ID

INS_REWK

A.1.8.3 Brief Description

During the rework stage, the author fixes the defects that were recorded during the inspection meeting.

A.1.8.4 Overview Description

<Field is Empty>

A.1.8.5 Summary Description

<Field is Empty>

A.1.8.6 Parent Template(s)

INS

A.1.8.7 Child Templates...

<Fields are Empty>

A.1.8.8 Used Within...**A.1.8.8.1 Output Object Name**

Peer_Review_Procedure_Manual

A.1.8.8.1.1 Output Object Type

Procedure_Manual

A.1.8.8.1.2 Include In Next Build

Yes

A.1.8.9 Transition Text...

<Fields are Empty>

A.1.8.10 Activity Criteria and Process...

A.1.8.10.1 Entry Criteria

The inspection stage has been completed.

A.1.8.10.2 Internal Process

Within 1 working day of the inspection meeting, the moderator provides the author with a copy of the Inspection Error Log and Trivial Error Logs.

The moderator gets an estimate of rework from the author. This includes the time required and a commitment for a completion date from the author.

Within 2 working days after the inspection meeting, the moderator completes an Inspection Summary Report and generates an Inspection Meeting Exit Memo.

The moderator distributes a copy of the Inspection Meeting Exit Memo and the Inspection Summary Report to the inspection participants, the project manager, and the Quality Assurance Administrator.

The author completes rework of the requirements, design, or code. The author corrects the defects recorded in the Inspection Error Log by modifying the requirements, design, or code, as appropriate. In situations where the correction for the defect is not obvious, the author should request an informal review (i.e., walkthrough) or confer with the project manager or technical lead.

If a reinspection is not necessary, the author contacts the moderator to schedule an informal review of the reworked material. Otherwise, the author notifies the moderator that the reinspection can be scheduled.

A.1.8.10.3 Exit Criteria

The rework time and completion date have been obtained from the author.

The Inspection Meeting Exit Memo and Inspection Summary Report are complete and have been distributed by the moderator.

The rework is completed by the author.

The informal review of the reworked material has been scheduled with the moderator or the reinspection has been scheduled. If a reinspection was indicated, the process continues with the planning stage.

A.1.8.10.4 Invariants

<Field is Empty>

A.1.8.11 Related Products...

A.1.8.11.1 Unique ID

IIP

A.1.8.11.1.1 Usage

Before Starting

A.1.8.11.1.2 Impact

Changed

A.1.8.11.1.3 Tailoring Options

Required

A.1.8.11.1.4 Elaboration Text

<Field is Empty>

A.1.8.12 Related Supports...

A.1.8.12.1 Unique ID

DEV

A.1.8.12.1.1 Usage

Before Starting

A.1.8.12.1.2 Tailoring Options

Required

A.1.8.12.1.3 Elaboration Text

<Field is Empty>

A.1.9 ACTIVITY TEMPLATE: INSPECTION FOLLOW-UP

A.1.9.1 Name

Inspection Follow-Up

A.1.9.2 Unique ID

INS_FWUP

A.1.9.3 Brief Description

The moderator verifies that all the rework has been completed during the follow-up stage. If the moderator recorded a large number of significant problems in the material during the inspection meeting, he schedules a reinspection of the reworked material. Otherwise, this inspection is considered complete upon the moderator's verification of the modifications.

A.1.9.4 Overview Description

A.1.9.5 Summary Description

<Field is Empty>

A.1.9.6 Parent Template(s)

INS

A.1.9.7 Child Templates...

<Fields are Empty>

A.1.9.8 Used Within...

A.1.9.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.9.8.1.1 Output Object Type

Procedure_Manual

A.1.9.8.1.2 Include In Next Build

Yes

A.1.9.9 Transition Text...

<Fields are Empty>

A.1.9.10 Activity Criteria and Process...

A.1.9.10.1 Entry Criteria

The rework stage has been completed.

Reinspection was not indicated.

A.1.9.10.2 Internal Process

The moderator and the author review the reworked material. The moderator makes an approval decision on the reworked material.

Any remaining open issues that cannot be resolved by the author are documented in the Software Problem Reporting (SPR) system. The author assumes responsibility for ensuring that these issues are eventually resolved.

If the rework and open issues, if any, are approved, the moderator generates an Inspection Close Memo within 2 working days after the review with the author. This memo identifies the SPR problem number that was opened to record any remaining issues.

The moderator distributes a copy of the Inspection Close Memo to the inspection participants, the project manager, and the Quality Assurance Administrator. (Note that the Inspection Close Memo may be generated electronically.)

A.1.9.10.3 Exit Criteria

The Inspection Close Memo has been completed and distributed.

A.1.9.10.4 Invariants

<Field is Empty>

A.1.9.11 Related Products...

A.1.9.11.1 Unique ID

IIP

A.1.9.11.1.1 Usage

Before Starting

A.1.9.11.1.2 Impact

Changed

A.1.9.11.1.3 Tailoring Options

Required

A.1.9.11.1.4 Elaboration Text

<Field is Empty>

A.1.9.11.2 Unique ID

RMET

A.1.9.11.2.1 Usage

Before Starting

A.1.9.11.2.2 Impact

Created

A.1.9.11.2.3 Tailoring Options

Required

A.1.9.11.2.4 Elaboration Text

<Field is Empty>

A.1.9.12 Related Supports...

A.1.9.12.1 Unique ID

DEV

A.1.9.12.1.1 Usage

Before Starting

A.1.9.12.1.2 Tailoring Options

Required

A.1.9.12.1.3 Elaboration Text

<Field is Empty>

A.1.9.12.2 Unique ID

MOD

A.1.9.12.2.1 Usage

Before Starting

A.19.12.2.2 Tailoring Options

Required

A.19.12.2.3 Elaboration Text

<Field is Empty>

A.1.10 ACTIVITY TEMPLATE: CAUSAL ANALYSIS

A.1.10.1 Name

Causal Analysis

A.1.10.2 Unique ID

CA

A.1.10.3 Brief Description

The purpose of the Causal Analysis activity is to identify and remove the cause of defects.

A.1.10.4 Overview Description

The Causal Analysis activity consists of a meeting attended by anyone who has participated in the inspection process. The purpose of the meeting is to make recommendations with regards to how processes may be improved in the sense of reducing the potential for and occurrence of defects in products. The scope of this process improvement includes not only all related engineering processes, but also the inspection process itself.

The causal analysis stage is a recent addition to the original method and is oriented toward improving the development process. Following the inspection meeting, the moderator holds a causal analysis meeting to encourage the inspection participants to identify root causes for defects which occurred with high frequency in the inspection material. Once they identify the root causes, the moderator solicits suggestions for process improvements which eliminate these root causes. The project team uses these suggestions to improve the development process. This meeting is typically limited to 30 minutes.

A.1.10.5 Summary Description

Meetings are scheduled to be between 1 and 2 hours in duration and within a given division, should be held at least monthly.

A.1.10.6 Parent Template(s)

<Field is Empty>

A.1.10.7 Child Templates...

<Fields are Empty>

A.1.10.8 Used Within...

A.1.10.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.1.10.8.1.1 Output Object Type

Procedure_Manual

A.1.10.8.1.2 Include In Next Build

Yes

A.1.10.9 Transition Text...

<Fields are Empty>

A.1.10.10 Activity Criteria and Process...**A.1.10.10.1 Entry Criteria**

Metrics are available from the inspection process that provide insight into the types of defects occurring in products, their frequency, and their severity.

The review coordinator has received a sufficient number of confirmed responses to justify holding the meeting.

A.1.10.10.2 Internal Process

The first part of the causal analysis meeting consists of examining metrics from the inspection activities and identifying the types of defects that occur with the highest frequency (or which, given their frequency, have the greatest undesirable impact). Using the technique of "foundation cause analysis" the group strives to identify process or environmental changes which can be made with the objective of removing some or all of the factors that contribute to that type of defect.

A.1.10.10.3 Exit Criteria

Minutes of the meeting are documented.

Action items have been reviewed.

A.1.10.10.4 Invariants

<Field is Empty>

A.1.10.11 Related Products...**A.1.10.11.1 Unique ID**

RMET

A.1.10.11.1.1 Usage

Before Starting

A.1.10.11.1.2 Impact

Left Unchanged

A.1.10.11.1.3 Tailoring Options

Required

A.1.10.11.1.4 Elaboration Text

<Field is Empty>

A.1.10.11.2 Unique ID

MEMO_RES

A.1.10.11.2.1 Usage

Before Starting

A.1.10.11.2.2 Impact

Left Unchanged

A.1.10.11.2.3 Tailoring Options

Suggested

A.1.10.11.2.4 Elaboration Text

<Field is Empty>

A.1.10.12 Related Supports...

A.1.10.12.1 Unique ID

PRT

A.1.10.12.1.1 Usage

While Underway

A.1.10.12.1.2 Tailoring Options

Required

A.1.10.12.1.3 Elaboration Text

<Field is Empty>

A.1.10.12.2 Unique ID

RCO

A.1.10.12.2.1 Usage

Before Starting

A.1.10.12.2.2 Tailoring Options

Required

A.1.10.12.2.3 Elaboration Text

<Field is Empty>

A.2 PEER REVIEW PRODUCT TEMPLATES

A.2.1 PRODUCT TEMPLATE: INSPECTION INPUT PRODUCT**A.2.1.1 Name**

Inspection Input Product

A.2.1.2 Unique ID

IIP

A.2.1.3 Brief Description

This is the product, item, or artifact to be inspected.

A.2.1.4 Overview Description

The inspection input product is anything that can be subjected to static analysis. This includes, but is not limited to, software requirements documents, software design documents, software code, and essentially all other types of documentation, including policies, procedures, standards, proposals, etc.

A.2.1.5 Summary Description

<Field is Empty>

A.2.1.6 Parent Template(s)

<Field is Empty>

A.2.1.7 Child Templates...

<Fields are Empty>

A.2.1.8 Used Within...**A.2.1.8.1 Output Object Name**

Peer_Review_Procedure_Manual

A.2.1.8.1.1 Output Object Type

Procedure_Manual

A.2.1.8.1.2 Include In Next Build

Yes

A.2.1.9 Transition Text...

<Fields are Empty>

A.2.1.10 Related Activities

INS

A.2.2 PRODUCT TEMPLATE: REVIEW MEMOS

A.2.2.1 Name

Review Memos

A.2.2.2 Unique ID

MEMO

A.2.2.3 Brief Description

This product is the set of memorandums that are used to support the inspection process.

A.2.2.4 Overview Description

The inspection process is facilitated by the use of memorandums. These serve three major purposes: (1) to convey information about upcoming inspection activities, (2) to serve as a type of checklist for those participating in the inspection process, (3) as an informal means to temporarily report metric information.

There are two primary types of memorandums currently in use: (1) The Invitation memorandum, and (2) the Results memorandum.

A.2.2.5 Summary Description

In addition to these two types of memorandums, moderators and review coordinators are encouraged to develop and use other memorandums if it appears that they will improve the efficiency or effectiveness of the inspection process.

If additional memorandums are developed, they should be taken to the causal analysis meeting for consideration and possible recommendation as new inspection support products.

A.2.2.6 Parent Template(s)

<Field is Empty>

A.2.2.7 Child Templates...

A.2.2.7.1 Unique ID

MEMO_INV

A.2.2.7.1.1 Child Type

Specialization

A.2.2.7.1.2 Child Tailoring

Recommended

A.2.2.7.1.3 Child Proximity

Local

A.2.2.7.2 Unique ID

MEMO_REV

A.2.2.7.2.1 Child Type

Specialization

A.2.2.7.2.2 Child Tailoring

Recommended

A.2.2.7.2.3 Child Proximity

Local

A.2.2.8 Used Within...**A.2.2.8.1 Output Object Name**

Peer_Review_Procedure_Manual

A.2.2.8.1.1 Output Object Type

Procedure_Manual

A.2.2.8.1.2 Include In Next Build

Yes

A.2.2.9 Transition Text...**A.2.2.9.1 Output Object Name**

Peer_Review_Procedure_Manual

A.2.2.9.1.1 Leading Transition Text

<Field is Empty>

A.2.2.9.1.2 Trailing Transition Text

Following are specific details about these two types of memorandums.

A.2.2.10 Related Activities

INS

A.2.3 PRODUCT TEMPLATE: INSPECTION INVITATION MEMORANDUM

A.2.3.1 Name

Inspection Invitation Memorandum

A.2.3.2 Unique ID

MEMO_INV

A.2.3.3 Brief Description

This memorandum is used to inform participants of an upcoming inspection.

A.2.3.4 Overview Description

The Invitation memorandum informs all participants that they have been selected to participate in a coming inspection. The memorandum includes the name of the artifact to be inspected, the amount of time the reviewer is expected to spend preparing for the inspection, the date that the inspection material will be distributed, the date and time of the inspection, and the role each person in the review team is expected to play.

A.2.3.5 Summary Description

<Field is Empty>

A.2.3.6 Parent Template(s)

MEMO

A.2.3.7 Child Templates...

<Fields are Empty>

A.2.3.8 Used Within...

A.2.3.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.2.3.8.1.1 Output Object Type

Procedure_Manual

A.2.3.8.1.2 Include In Next Build

Yes

A.2.3.9 Transition Text...

A.2.3.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.2.3.9.1.1 Leading Transition Text

<Field is Empty>

A.2.3.9.1.2 Trailing Transition Text

For an example of what this memorandum looks like, refer to the appendixes.

A.2.3.10 Related Activities

INS_PLAN

A.2.4 PRODUCT TEMPLATE: INSPECTION RESULTS MEMORANDUM

A.2.4.1 Name

Inspection Results Memorandum

A.2.4.2 Unique ID

MEMO_RES

A.2.4.3 Brief Description

This memorandum is used to report the results of a completed Inspection activity.

A.2.4.4 Overview Description

The Results memorandum is used to collect and summarize information about a completed inspection. The primary purpose of this memorandum is to serve as a vehicle for capturing inspection information that may or may not be transferred into an automated information management system or database. In any event, this memorandum serves as a permanent record of the inspection and its results.

A.2.4.5 Summary Description

<Field is Empty>

A.2.4.6 Parent Template(s)

MEMO

A.2.4.7 Child Templates...

<Fields are Empty>

A.2.4.8 Used Within...

A.2.4.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.2.4.8.1.1 Output Object Type

Procedure_Manual

A.2.4.8.1.2 Include In Next Build

Yes

A.2.4.9 Transition Text...

A.2.4.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.2.4.9.1.1 Leading Transition Text

<Field is Empty>

A.2.4.9.1.2 Trailing Transition Text

For an example of what this memorandum looks like, refer to the appendixes.

A.2.4.10 Related Activities

INS_PLAN

A.2.5 PRODUCT TEMPLATE: REVIEW METRICS

A.2.5.1 Name

Review Metrics

A.2.5.2 Unique ID

RMET

A.2.5.3 Brief Description

This is a hard-copy report summarizing the inspection results.

A.2.5.4 Overview Description

The review metrics product is a report that contains all the metric information gathered during the inspection process. The primary contents of this report includes:

- Number of defects (by type and severity) found during preparation
- Number of defects (by type and severity) found during the inspection meeting
- Number of people that prepared for the inspection
- Total time spent in preparation
- Number of people that attended the inspection meeting
- Inspection meeting start time
- Inspection meeting completion time
- Total staff-time spent reviewing the product within the inspection meeting
- Estimated rework time
- Actual rework time
- Names and roles of all participants

A.2.5.5 Summary Description

<Field is Empty>

A.2.5.6 Parent Template(s)

<Field is Empty>

A.2.5.7 Child Templates...

<Fields are Empty>

A.2.5.8 Used Within...**A.2.5.8.1 Output Object Name**

Peer_Review_Procedure_Manual

A.2.5.8.1.1 Output Object Type

Procedure_Manual

A.2.5.8.1.2 Include In Next Build

Yes

A.2.5.9 Transition Text...**A.2.5.9.1 Output Object Name**

Peer_Review_Training_Material

A.2.5.9.1.1 Leading Transition Text

<Field is Empty>

A.2.5.9.1.2 Trailing Transition Text

On a separate piece of paper, describe at least five additional metrics that might be important to collect during the inspection process. Explain both how and when they would be collected and for what they would be used. Please spend about 20 minutes on this exercise.

A.2.5.10 Related Activities

INA

A.3 PEER REVIEW ROLE TEMPLATES

A.3.1 ROLE TEMPLATE: PEER REVIEW TEAM**A.3.1.1 Name**

Peer Review Team

A.3.1.2 Unique ID

PRT

A.3.1.3 Brief Description

The Peer Review Team participates in and is responsible for all phases of the Peer Review process.

A.3.1.4 Overview Description

Peer Review Teams conduct reviews of products (sometimes several times) before those products are placed into Configuration Management. The team typically contains three to six people who have one or more of the roles of: Moderator, Scribe, Reader, Key and Regular Inspectors, and Developer. Note that the developer of the product being inspected serves as part of the review team.

A.3.1.5 Summary Description

<Field is Empty>

A.3.1.6 Parent Template(s)

<Field is Empty>

A.3.1.7 Child Templates...**A.3.1.7.1 Unique ID**

PRT_MOD

A.3.1.7.1.1 Child Type

Composition

A.3.1.7.1.2 Child Tailoring

Required

A.3.1.7.1.3 Child Proximity

Local

A.3.1.7.2 Unique ID

PRT_RDR

A.3.1.7.2.1 Child Type

Composition

A.3.1.7.2.2 Child Tailoring

Required

A.3.1.7.2.3 Child Proximity

Local

A.3.1.7.3 Unique ID

PRT_SCB

A.3.1.7.3.1 Child Type

Composition

A.3.1.7.3.2 Child Tailoring

Required

A.3.1.7.3.3 Child Proximity

Local

A.3.1.7.4 Unique ID

PRT_INS

A.3.1.7.4.1 Child Type

Composition

A.3.1.7.4.2 Child Tailoring

Required

A.3.1.7.4.3 Child Proximity

Local

A.3.1.7.5 Unique ID

PRT_DEV

A.3.1.7.5.1 Child Type

Composition

A.3.1.7.5.2 Child Tailoring

Suggested

A.3.1.7.5.3 Child Proximity

Local

A.3.1.8 Used Within...

A.3.1.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.1.8.1.1 Output Object Type

Procedure_Manual

A.3.1.8.1.2 Include In Next Build

Yes

A.3.1.9 Transition Text...

A.3.1.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.1.9.1.1 Leading Transition Text

<Field is Empty>

A.3.1.9.1.2 Trailing Transition Text

Details about each of these roles are presented below.

A.3.1.10 Supported Activities

INA

CA

A.3.1.11 Reports To...

<Not Applicable>

A.3.1.12 Reported To By...

<Not Applicable>

A.3.2 ROLE TEMPLATE: MODERATOR

A.3.2.1 Name

Moderator

A.3.2.2 Unique ID

PRT_MOD

A.3.2.3 Brief Description

The moderator manages the overall inspection process and ensures that the requirements and intent of the inspection method are met.

A.3.2.4 Overview Description

The moderator is responsible for all aspects of a particular review process. Once selected as moderator, that person sees that the review process is conducted as efficiently and effectively as possible.

A.3.2.5 Summary Description

<Field is Empty>

A.3.2.6 Parent Template(s)

PRT

A.3.2.7 Child Templates...

<Fields are Empty>

A.3.2.8 Used Within...

A.3.2.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.2.8.1.1 Output Object Type

Procedure_Manual

A.3.2.8.1.2 Include In Next Build

Yes

A.3.2.9 Transition Text...

A.3.2.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.2.9.1.1 Leading Transition Text

<Field is Empty>

A.3.2.9.1.2 Trailing Transition Text

<Field is Empty>

A.3.2.10 Supported Activities

INA_PLAN

INA_OVR

INA_PREP

INA_MTG

INA_REWK

INA_FWUP

A.3.2.11 Reports To...**A.3.2.11.1 Unique ID**

RCO

A.3.2.11.1.1 Elaboration Text

After every inspection, the moderator reports the results to the review coordinator. Additionally, the moderator has access to the review coordinator at any time for advice, recommendations, etc.

A.3.2.12 Reported To By...

MOD

SCB

RDR

INS

DEV

A.3.3 ROLE TEMPLATE: SCRIBE

A.3.3.1 Name

Scribe

A.3.3.2 Unique ID

PRT_SCB

A.3.3.3 Brief Description

The scribe records and classifies the defects identified during the inspection meeting.

A.3.3.4 Overview Description

The scribe role only exists within the actual inspection meeting itself. During the meeting the scribe is responsible for building the defect log. As the meeting progresses, the scribe notes the defects identified by type, location, and severity. At the end of the meeting, the scribe gives the defect log to the moderator.

A.3.3.5 Summary Description

<Field is Empty>

A.3.3.6 Parent Template(s)

PRT

A.3.3.7 Child Templates...

<Fields are Empty>

A.3.3.8 Used Within...

A.3.3.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.3.8.1.1 Output Object Type

Procedure_Manual

A.3.3.8.1.2 Include In Next Build

Yes

A.3.3.9 Transition Text...

A.3.3.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.3.9.1.1 Leading Transition Text

<Field is Empty>

A.3.3.9.1.2 Trailing Transition Text

<Field is Empty>

A.3.3.10 Supported Activities

INS_MTG

A.3.3.11 Reports To...

A.3.3.11.1 Unique ID

PRT_MOD

A.3.3.11.1.1 Elaboration Text

<Field is Empty>

A.3.3.12 Reported To By...

<Field is Empty>

A.3.4 ROLE TEMPLATE: READER

A.3.4.1 Name

Reader

A.3.4.2 Unique ID

PRT_RDR

A.3.4.3 Brief Description

The reader presents the product during the inspection meeting.

A.3.4.4 Overview Description

The reader is responsible for presenting the material to be inspected during the inspection meeting. This presentation consists of advancing through the material in a manner that allows the participants of the meeting to present any defects they had previously identified and to again review the material for as yet undetected defects.

A.3.4.5 Summary Description

<Field is Empty>

A.3.4.6 Parent Template(s)

PRT

A.3.4.7 Child Templates...

<Fields are Empty>

A.3.4.8 Used Within...

A.3.4.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.4.8.1.1 Output Object Type

Procedure_Manual

A.3.4.8.1.2 Include In Next Build

Yes

A.3.4.9 Transition Text...

A.3.4.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.4.9.1.1 Leading Transition Text

<Field is Empty>

A.3.4.9.1.2 Trailing Transition Text

<Field is Empty>

A.3.4.10 Supported Activities

INS_OVR

INS_MTG

A.3.4.11 Reports To...

A.3.4.11.1 Unique ID

PRT_MOD

A.3.4.11.1.1 Elaboration Text

<Field is Empty>

A.3.4.12 Reported To By...

<Field is Empty>

A.3.5 ROLE TEMPLATE: INSPECTORS

A.3.5.1 Name

Inspectors

A.3.5.2 Unique ID

PRT_INS

A.3.5.3 Brief Description

Inspectors evaluate material against both formal and informal standards to identify and log defects.

A.3.5.4 Overview Description

Each inspector is responsible for reviewing the material during the preparation stage and for participating in the inspection meeting. Their goal during the preparation activity and the inspection meeting is to find defects in the artifact or material being inspected. Inspectors are considered to be of two types: key inspectors and regular inspectors.

A.3.5.5 Summary Description

In addition to key and regular inspectors there is also the informal role of "guest inspector." Although rarely part of the process, a guest inspector is someone who does not participate in any of the meetings. Instead, guest inspectors evaluate the product and send the results of their evaluation to the moderator.

A.3.5.6 Parent Template(s)

PRT

A.3.5.7 Child Templates...

A.3.5.7.1 Unique ID

PRT_INS_KEY

A.3.5.7.1.1 Child Type

Specialization

A.3.5.7.1.2 Child Tailoring

Required

A.3.5.7.1.3 Child Proximity

Local

A.3.5.7.2 Unique ID

PRT_INS_REG

A.3.5.7.2.1 Child Type

Specialization

A.3.5.7.2.2 Child Tailoring

Suggested

A.3.5.7.2.3 Child Proximity

Local

A.3.5.8 Used Within...

A.3.5.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.5.8.1.1 Output Object Type

Procedure_Manual

A.3.5.8.1.2 Include In Next Build

Yes

A.3.5.9 Transition Text...

A.3.5.9.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.5.9.1.1 Leading Transition Text

<Field is Empty>

A.3.5.9.1.2 Trailing Transition Text

<Field is Empty>

A.3.5.9.2 Output Object Name

Peer_Review_Training_Material

A.3.5.9.2.1 Leading Transition Text

<Field is Empty>

A.3.5.9.2.2 Trailing Transition Text

On a separate piece of paper, briefly describe one or two additional types of inspectors the might be useful to have in the Peer Review process. Please spend about 10 minutes on this exercise.

A.3.5.10 Supported Activities

INS_OVR
INS_PREP
INS_MTG

A.3.5.11 Reports To...

A.3.5.11.1 Unique ID

PRT_MOD

A.3.5.11.1.1 Elaboration Text

<Field is Empty>

A.3.5.12 Reported To By...

<Field is Empty>

A.3.6 ROLE TEMPLATE: DEVELOPER**A.3.6.1 Name**

Developer

A.3.6.2 Unique ID

PRT_DEV

A.3.6.3 Brief Description

The developer is the person currently responsible for working on a product.

A.3.6.4 Overview Description

A developer is anyone who is currently assigned to, working on, creating, or is otherwise responsible for a product. It is often the case that a team of developers share responsibility for the development of a product. In such circumstances, one person on that team has the role of lead developer.

A.3.6.5 Summary Description

<Field is Empty>

A.3.6.6 Parent Template(s)

PRT

A.3.6.7 Child Templates...

<Fields are Empty>

A.3.6.8 Used Within...**A.3.6.8.1 Output Object Name**

Peer_Review_Procedure_Manual

A.3.6.8.1.1 Output Object Type

Procedure_Manual

A.3.6.8.1.2 Include In Next Build

Yes

A.3.6.9 Transition Text...**A.3.6.9.1 Output Object Name**

Peer_Review_Procedure_Manual

A.3.6.9.1.1 Leading Transition Text

<Field is Empty>

A.3.6.9.1.2 Trailing Transition Text

<Field is Empty>

A.3.6.10 Supported Activities

INS_PLAN

INS_OVR

INS_PREP

INS_MTG

INS_REWK

INS_FWUP

A.3.6.11 Reports To...

A.3.6.11.1 Unique ID

PRT_MOD

A.3.6.11.1.1 Elaboration Text

<Field is Empty>

A.3.6.12 Reported To By...

<Field is Empty>

A.3.7 ROLE TEMPLATE: KEY INSPECTOR**A.3.7.1 Name**

Key Inspector

A.3.7.2 Unique ID

PRT_INS_KEY

A.3.7.3 Brief Description

Key inspectors are inspectors that are required at the inspection meeting.

A.3.7.4 Overview Description

Key inspectors have the primary responsibility for inspecting the product or artifact under review. Generally, candidates are identified to be key inspectors due to particularly high levels of experience, insight, or ability that they can contribute to reviewing the artifact. Generally, key inspectors should have different specialty areas so as to provide greater scope in evaluating the artifact.

A.3.7.5 Summary Description

<Field is Empty>

A.3.7.6 Parent Template(s)

PRT_INS

A.3.7.7 Child Templates...

<Fields are Empty>

A.3.7.8 Used Within...**A.3.7.8.1 Output Object Name**

Peer_Review_Procedure_Manual

A.3.7.8.1.1 Output Object Type

Procedure_Manual

A.3.7.8.1.2 Include In Next Build

Yes

A.3.7.9 Transition Text...

<Fields are Empty>

A.3.7.10 Supported Activities

<Field is Empty>

A.3.7.11 Reports To...

<Fields are Empty>

A.3.7.12 Reported To By

<Field is Empty>

A.3.8 ROLE TEMPLATE: REGULAR INSPECTOR

A.3.8.1 Name

Regular Inspector

A.3.8.2 Unique ID

PRT_INS_REG

A.3.8.3 Brief Description

Regular inspectors are inspectors that are optional at the inspection meeting.

A.3.8.4 Overview Description

Regular inspectors are inspectors who are expected to participate throughout the inspection process but who will not cause the cancellation of an inspection meeting if they cannot attend. All inspectors start as regular inspectors and, after they have acquired sufficient experience, may periodically be selected to be key inspectors.

A.3.8.5 Summary Description

<Field is Empty>

A.3.8.6 Parent Template(s)

PRT_INS

A.3.8.7 Child Templates...

<Fields are Empty>

A.3.8.8 Used Within...

A.3.8.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.8.8.1.1 Output Object Type

Procedure_Manual

A.3.8.8.1.2 Include In Next Build

Yes

A.3.8.9 Transition Text...

<Fields are Empty>

A.3.8.10 Supported Activities

<Field is Empty>

A.3.8.11 Reports To...

<Fields are Empty>

A.3.8.12 Reported To By

<Field is Empty>

A.3.9 ROLE TEMPLATE: REVIEW COORDINATOR

A.3.9.1 Name

Review Coordinator

A.3.9.2 Unique ID

RCO

A.3.9.3 Brief Description

Review coordinators are responsible for organizing and overseeing reviews.

A.3.9.4 Overview Description

The primary responsibility of review coordinators is to see that all planned reviews are conducted according to schedule, that the necessary output products are produced and archived, and that reviews are conducted in an efficient and effective manner. One key area of responsibility is ensuring that review work is evenly and fairly divided among all those who are authorized to participate. As a secondary responsibility, review coordinators also ensure that all participants have the necessary training, or if not, that arrangements are made so that training occurs.

A.3.9.5 Summary Description

<Field is Empty>

A.3.9.6 Parent Template(s)

<Field is Empty>

A.3.9.7 Child Templates...

A.3.9.8 Used Within...

A.3.9.8.1 Output Object Name

Peer_Review_Procedure_Manual

A.3.9.8.1.1 Output Object Type

Procedure_Manual

A.3.9.8.1.2 Include In Next Build

Yes

A.3.9.9 Transition Text...

<Fields are Empty>

A.3.9.10 Supported Activities

INS

CA

A.3.9.11 Reports To...

<Fields are Empty>

A.3.9.12 Reported To By

PRT_MOD

APPENDIX B. CHECKLISTS

This Appendix contains checklists that you can adapt to help you review your process definition work. Consider using these checklists for inspections of filled in templates.

The checklists include a statement, question or claim, followed by two boxes. For each statement put a value in the first box that ranges from -3 to 3, where -3 indicates Definitely False or Definitely Disagree and where 3 indicates Definitely True or Definitely Agree. Of course, 0 indicates you are neutral or have no opinion, 1 and -1 indicate you slightly agree or slightly disagree, respectively; etc.

For each statement, in the second box put a value of L, M, or H indicating Low, Medium, or High confidence level in your numeric rating.

For example, upon reading the statement, "Are all portions of the template filled in that should be?" you might respond with a 2 in the first box and a L in the second box. This reflects, for example, that you agree with the statement, based on what you have seen of the templates; but you have only seen a few of the templates so, at this time, your confidence is low. Later, upon reviewing more templates you would likely increase your confidence to M or H, and possibly change your numeric rating as appropriate.

You are encouraged to reduce the following checklists to something that you and your peers can use to support review and evaluation of your process definition work. You will find that even a very brief checklist can considerably simplify, standardize, and accelerate reviews, evaluations, and inspections of your process definition.

The following areas and categories are addressed:

ACTIVITY TEMPLATE

- Elementary
- Advanced

ATTRACTIVENESS FOR ADOPTION

- Introduction
- Business

ROLE USAGE

- Evaluators, Champions, and Change Agents
- Users, Learners, Enactors
- Tailor and Instantiator

DEFINITION PROPERTIES

- Scope and Power
- General Characteristics
- Performance Factors
- Global Characteristics Support

ADAPTABILITY

- Adaptability Factors

DETAILED CONTENT AND SUPPORT

- Startup Requirements
- Input Requirements
- Process and Method Architecture Requirements
- Body Requirements
- Output Requirements
- Data Requirements
- General Definition Requirements
- Method Description Desired/Optional Characteristics
- Kinds of Engineering Knowledge
- Methods Conformance
- Level of Detail
- Compatibility/Interoperability
- Level of integration
- Planned Improvements
- Method Automation
- Scope and Availability of Supporting Items
- Style and Aesthetics
- Role Usage Summary

COMPETITION

- Benchmark Information
- Definition Competition Information

B.1 ACTIVITY TEMPLATE*Category: Elementary*

#	Item	Rating -3 to +3	Confidence L, M, or H
1.	Are portions filled in that should be?		
2.	Does all terminology conform to glossary?		
3.	Do all names and identifiers conform to standards?		
4.	Is this activity described using means and terms that are the same as those used to describe other parts of the process?		
5.	Is this activity consistent with descriptions of other parts of process that it is supposed to be consistent with (e.g., parents, children, referenced in description, need to be composed or integrated with)?		
6.	Is scope of activity clear?		
7.	Is purpose of activity clear?		
8.	Is the description clear?		
9.	Are external references such that they can be easily obtained?		
10.	To the extent it is supposed to, does the description match the as-is process?		
11.	Is the description realistic?		
12.	Can all the users of the description understand it?		
13.	Does the description include enough so the activity can be done?		
14.	Does it contain the information needed for management planning and accounting (e.g., work breakdown structure element identification)?		
15.	Are all input and output artifacts identified?		
16.	Are important input and output non-artifact capabilities and knowledge identified (e.g., skills resulting from training)?		
17.	Does the user know where to go next for info/help?		
18.	Are internal artifacts identified?		
19.	Are common or severe pitfalls and risks identified?		
20.	Does the activity include verification?		

Category: Advanced

#	Item	Rating -3 to +3	Confidence L, M, or H
21.	Are all formal portions filled in that should be?		
22.	Are limitations clear?		
23.	Does it conform to process architecture?		
24.	Are states of the activity and transition criteria identified?		

Category: Advanced (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
25.	Is timing covered?		
26.	Is tailoring covered?		
27.	Are contingencies and responses (e.g., cases, input changes, exceptions, rework) identified?		
28.	Are organizational aspects covered?		
29.	Are starting and ending dealt with in addition to input and output conditions/criteria?		
30.	Are options and alternatives covered?		
31.	Is validation included?		
32.	Is measurement included?		
33.	Is quality assurance included?		
34.	Is risk assessment included?		
35.	Is management of this activity covered?		
36.	Is process improvement covered?		
37.	Is opportunity identification and exploitation covered?		
38.	Is automation covered?		
39.	Is reuse covered?		
40.	Does it provide advanced, sound, and complete guidance?		
41.	Is it better than competition?		
42.	Is it easily adaptable?		
43.	Is it easily evolvable?		

B.2 GENERAL PROCESS DEFINITION CHECKLIST

This section of the appendix provides definition checklists for each of the following categories:

1. ***Attractiveness for Adoption.*** Exhibits the characteristics that usually encourage a decision to transfer/adopt.
2. ***Role Usage.*** Ease of successful use by each different user type (e.g., evaluators, tailors, learners, and enactors).
3. ***Definition Properties.*** Exhibits those characteristics that maximize use and usefulness.
4. ***Adaptability.*** Suitability for tailoring and evolving all or part—highlights this important property.
5. ***Detailed Contents.*** Contents are complete, including automation considerations, and adequate support devices are in place.
6. ***Competition.*** Competition's comparative strengths and weaknesses are understood, and processes or methods and their definitions have been measured against their competition.

B.2.1 ATTRACTIVENESS FOR ADOPTION*Category: Introduction*

#	Item	Rating -3 to +3	Confidence L, M, or H
1.	Concern for your organization's transition from existing practices to the newly developed process or method is addressed. Concern for transition from existing releases is incorporated into each upgrade.		
2.	Definition is compatible with existing organizational requirements.		
3.	All or part of the definition can be used on a trial basis without permanently impacting existing techniques (trialability).		
4.	Definition can be adopted piecemeal (modularity).		
5.	Definition can be used on a trial basis to ensure applicability and cost-effectiveness for organization (trialability).		
6.	Effects are easily observable.		

Category: Business

#	Item	Rating -3 to +3	Confidence L, M, or H
7.	Credibility of benefits is high.		
8.	Has prior use.		
9.	Has prior evaluation results that are quantitative and qualitative and show realism and credibility.		
10.	Conforms to standards.		
11.	Original sources have strong reputation, history, and stability.		
12.	Has testing and verification history. Availability of results, independent verification and validation, and thoroughness.		
13.	The implementation satisfies cost-benefit criteria (which includes product quality considerations).		
14.	Is predictable, reducing risks taken by customer.		
15.	Has indirect benefits of ease of learning and positive market impact.		
16.	Has low cost start-up.		
17.	Satisfies all items on Competition checklist.		

B.2.2 ROLE USAGE*Category:* Evaluators, Champions, and Change Agents

#	Item	Rating -3 to +3	Confidence L, M, or H
1.	Definition identifies an aggregate list of all stakeholders (those types of persons who will be affected by any decisions made during an instantiation of the definition).		
2.	Includes assessment of cost of introduction.		
3.	Includes assessment of cost of operation, maintenance, and use.		
4.	Includes assessment of cost of replacement.		
5.	Includes assessment of direct, expected benefit from the results (type, quality, quantity).		
6.	Includes assessment of time reduction benefits.		
7.	Includes assessment of resource reduction benefits (e.g., definition lets customer reduce technology and project costs).		
8.	Has prior demonstrations of use indicated.		
9.	Ease of installation and start-up is incorporated.		
10.	The definition has been packaged and productized such that it minimizes adoption work by organization.		

Category: Users, Learners, Enactors

#	Item	Rating -3 to +3	Confidence L, M, or H
11.	<p>The definition addresses the following user needs:</p> <ul style="list-style-type: none"> To know where the definition might be useful To quickly find part(s) that is needed or desired or establish that the part(s) does not exist To decide to use the part(s) To do detailed plan to use part(s) To know what is needed to use the part(s) Use the part(s) To know when he/she is straying from the intended usage and return to the right usage if desired To be able to recover when errors are encountered To know when he/she/they is finished To know the quality of the result 		
12.	Definition is relevant and appropriate.		
13.	Definition has logical qualities. Includes unifying idea that postulates nothing unnecessary, internally is logically consistent, is conceivably refutable, is explicitly bounded in application, and is consistent with previously established theories and laws.		

Category: Users, Learners, Enactors (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
14.	Definition has empirical qualities. Makes observationally confirmable predictions or conclusions, confirming observations are reproducible by skeptics, and provides criteria for observations as fact or anomaly.		
15.	Definition is usable and can be interacted with successfully by business area managers, project managers, line engineers, and technologists.		
16.	Definition is accurate. Is the document correct? Are there obvious mistakes? Are assumptions about resources and environment valid? Is there evidence of a lack of understanding of important aspects of the problem or process?		
17.	Definition is clear. Will it be understandable to the users? Is the document expressed in a form that is accessible and understandable? Are table and diagrams used where possible instead of text?		
18.	Definition is consistent. Definition possesses the characteristic of consistency to the extent the definition products correlate and contain uniform notation, terminology, and symbology. In addition, passages do not contradict each other.		
19.	Is the document organized so that it can serve different user populations simultaneously?		
20.	Allows learners to start immediately on meaningfully realistic tasks.		
21.	Reduces the amount of reading and other passive activity in learning.		
22.	Helps to make errors and error recovery less traumatic and more proactive in reducing future errors.		
23.	Is able to successfully deal with realistic range of your organization personnel skills and knowledge.		
24.	Includes guidance for resource and planning estimates.		

Category: Tailor and Instantiator

#	Item	Rating -3 to +3	Confidence L, M, or H
25.	Uses generic terminology and symbols for ease of adaptation.		
26.	Methods work in standalone mode as well as integrated within a process.		
27.	<p>Alternative tailorings and instantiations should support any needed variations for:</p> <ul style="list-style-type: none"> Different life-cycle models Levels of human/organizational capability Level of understanding of requirements or architecture Available reusable assets, including mix of reused and new components Tailoring at organization, project, team, individual levels Automated support differences, including mixed levels of automation Dependability criticality Performance criticality and real-time Data intensity Degree of distribution Size Customer types and standards (e.g., commercial, DoD, NASA, NSA, FAA, etc.) Programming language (e.g., Ada, C, or C++) Subsetting Likely changes/enhancements/evolution 		
28.	instantiation bindings should support organizational structure, physical communication, relative and then absolute time, staffing qualities and needs, and identities of other needed resources.		
29.	<p>Composition and integration of all or part of the definition should:</p> <ul style="list-style-type: none"> Match Ins and Outs Receive (re)direction Interface with other information flows <ul style="list-style-type: none"> Coordination, collaboration, informing, report to, measurements database, external sources of information and advice Have links fulfilling necessary dependency requirements 		

B.2.3 DEFINITION PROPERTIES*Category: Scope and Power*

#	Item	Rating -3 to +3	Confidence L, M, or H
1.	Addresses known need or big problem area.		
2.	Applicable to existing environment(s).		
3.	The definition explains what is currently known in the world for sufficient state of the practice, best state of the practice, or state of the art. Includes algorithmic or automated procedures, elimination of ambiguity, and ability to overcome obstacles within conventional framework.		
4.	The definition explains the limitations of its contents and points to where opportunities exist for improvement and enhancement. Includes understanding of what part(s) of definition are automatic and why, understanding of ambiguity and obstacles.		
5.	The definition provides techniques for addressing those novel areas that go beyond what is fully understood. Includes application of definition to novel tasks, tolerance of ambiguity, and redefinition of obstacles		

Category: General Characteristics

#	Item	Rating -3 to +3	Confidence L, M, or H
6.	Correctness The extent to which the definition conforms to its system specifications and standards, is free from defects, and satisfies its specified requirements.		
7.	Completeness A product possesses the characteristics of completeness to the extent that all of its parts are present and each of its parts is fully developed; all aspects addressed (definitions, theory, data, techniques, evaluation, integration, extension, comparison, application, artifacts, anomaly detection and recovery).		
8.	Self-descriptiveness The definition explains the implementation of its methods and procedures.		
9.	Simplicity The definition provides implementation on its components in the most understandable and regular manner.		

Category: General Characteristics (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
10.	Adequacy Degree to which the definition satisfies the required and optional needs of the member companies.		
11.	Usefulness Includes effective, functional, feasible, durable, operable, usable, useful, workable.		
12.	Definition satisfies those items in the Adaptability checklist.		

Category: Performance Factors

#	Item	Rating -3 to +3	Confidence L, M, or H
13.	Efficiency The extent to which the definition performs its intended functions requiring a minimum amount of resources.		
14.	Integrity Maintaining the validity and consistency of data and of the components within the definition.		
15.	Reliability The extent to which all or part of the definition can be expected to perform its intended functions in a satisfactory manner.		
16.	Survivability The extent to which the definition will perform and support critical functions without failures within a specified time period when a portion of the definition is inadequate, unsuitable, or inoperable.		
17.	The definition minimizes the amount of processing that needs to be done both during actual execution of a component and during transition from one component to another.		
18.	The definition minimizes the amount of data that needs to be stored during transition from one component to another.		

Category: Global Characteristics Support

#	Item	Rating -3 to +3	Confidence L, M, or H
19.	Scale up		
20.	Repeatability		
21.	Measurability		
22.	Integrability		
23.	Early prediction and validation of benefits		
24.	Easily learned		
25.	Cost-effective		

B.2.4 ADAPTABILITY*Category:* Adaptability Factors

#	Item	Rating -3 to +3	Confidence L, M, or H
1.	Modularity The extent to which the definition is composed of discrete components such that a change to one component has minimal impact on other components.		
2.	Reconfigurability The relative effort needed to change the interconnection and status of definition components and links.		
3.	Expandability/Extensibility The extent to which the definition is designed to facilitate development and use of extensions or expansions and the extent to which the definition is reusable so that the parts can be combined to create new interfaces and facilities.		
4.	Flexibility The relative effort to increase definition capability by enhancing current components or adding new components or data.		
5.	Portability The ease with which the definition can be transferred from one computer system or environment to another.		
6.	Reusability The extent to which all or part of the definition can be used in multiple applications or multiple process architectures.		
7.	Augmentability The relative effort for the customer to increase or adapt the functionality of the definition by adding new functions or modifying existing functions.		
8.	Commonality The extent to which there is commonality and reuse across the methods for multiple activities or process architectures.		
9.	Definition is customizable/tailorable.		
10.	Traceability Those characteristics of the definition that provide a thread of origin from the implementation to the requirements with respect to the specific development envelope and operational environment.		

Category: Adaptability Factors (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
11.	Maintainability The extent to which the definition facilitates updating to satisfy new requirements or to correct deficiencies. This implies that the definition is understandable, testable, and modifiable.		
12.	Interoperability The ability of components within the definition to exchange information and to mutually use the information that has been exchanged.		
13.	Generality Degree to which the definition is applicable in different applications, organizations, and environments.		
14.	Open architecture		
15.	Visibility Those characteristics of the definition which provide status monitoring of its development and use.		
16.	Verifiability The relative effort to verify the contents, operation and performance of the definition.		
17.	Definition satisfies those items in the Role Usage checklist under the Instantiator category.		

B.2.5 DETAILED CONTENTS*Category: Startup Requirements*

#	Item	Rating -3 to +3	Confidence L, M, or H
1.	States what the user is expected to know before using.		
2.	States where the method is applicable and any factors that must be taken into consideration for use.		
3.	Includes or references standard introductory material. Identification—includes name, description Status—maturity level, completeness, whether method represents best state of the art, best state of the practice or other Background—history Rationale and reasons for confidence Technical—concepts, definitions, notations		

Category: Input Requirements

#	Item	Rating -3 to +3	Confidence L, M, or H
4.	Input to method activity or includes input products, entrance criteria, standards and other governing documents and direction, operations/transactions to be handled by the activity or the method, and dependency requirements for method (which may vary with the alternative method being used).		
5.	Entrance criteria include data and states (musts versus desirables versus exceptions), humans and states, other resources, "verification" steps, repair steps/suggestions if entrance criteria fail, other applicability considerations, discussion of granularity/scopes of input and output.		

Category: Process and Method Architecture Requirements

#	Item	Rating -3 to +3	Confidence L, M, or H
6.	Process and methods are built to be fault tolerant. Includes damage confinement, early defect correction, repair and recovery, continuing service to other (sub)activities.		
7.	Process and method architectures are structured to facilitate change. Activities known only through externally visible, specific behavior.		

Category: Process and Method Architecture Requirements (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
8.	Process architecture identifies the weakest set of constraints needed to ensure successful execution (e.g., weakest set of pre- and post-conditions on each method).		
9.	Process architecture and methods provide commonality across initial development of software and evolution of existing software.		
10.	Processes and methods are built to ensure early defect detection and should pay attention to repair methods for defect correction.		

Category: Body Requirements

#	Item	Rating -3 to +3	Confidence L, M, or H
11.	Clearly identifies heuristic (or better) steps and decision points that need to be executed for successful completion.		
12.	Body of method (which may have alternatives) contains internal data and states; internal standards, conditions, and rules; goals; issues.		
13.	<p>Body includes description of components.</p> <p>Components (Steps/Tasks/Operations/Subtransactions)</p> <p>Subgoals</p> <p>Procedures</p> <p>Analyses, derivations, creations, models,</p> <p>Planning, prerequisites</p> <p>Decisions</p> <p>Subgoals, constraints/rules, input and involvement, data collection procedures, alternatives (with rationales), claims, questions, anticipated results, desirables/suggestions/heuristics, means of prediction and evaluation</p> <p>Reuse</p> <p>Validation</p> <p>Action item creation, etc.</p> <p>Progress checkpoints</p> <p>Constraints on sequencing and feedback</p> <p>Absolutes, decisions, exceptions</p> <p>Automation</p> <p>Roles</p> <p>Embellishments</p>		
14.	Body of method includes decision criteria or activity that leads to output products within expected range of values.		
15.	Includes dependability characteristics (identification and avoidance of pitfalls and risks, mitigation of risks).		

Category: Body Requirements (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
16.	Includes measurement needs. Progress measurement/tracking, measurement for improvement, entrance and exit criteria, measurements or a checklist for measuring progress through execution, quality measures.		
17.	Includes continuous improvement subactivity.		
18.	Includes information flow. Report to, coordination, collaboration, informing, metrics reporting, links to CM.		
19.	Identifies any links to human, dependability, and performance engineering.		
20.	Clearly specifies interfaces with system engineering, configuration management, and QA.		
21.	Provides planning information/estimation, including degree of predictability, resources and time, quality, and rates of improvement.		

Category: Output Requirements

#	Item	Rating -3 to +3	Confidence L, M, or H
22.	Output from activity or method includes output products, exit criteria, exceptions, deliveries to, and obligations.		
23.	Exit criteria include data and states, verification, approvals, and internal repair steps and suggestions if exit criteria fail.		

Category: Data Requirements

#	Item	Rating -3 to +3	Confidence L, M, or H
24.	For data affected by a method, method addresses its: Consistency (attainment, preservation/propagation, inconsistency tolerance, verification, and repair) Completeness (attainment, incompleteness tolerance, verification, validation, and repair) Quality—suggested qualities (measures [Gilb columns], attainment, early defect detection, repair) Intermediate conditions/states		

Category: General Definition Requirements

#	Item	Rating -3 to +3	Confidence L, M, or H
25.	Examples are provided in the definition.		
26.	Definition identifies all stakeholders within each component (those who will be affected by any decisions made during enactment of that component).		
27.	Definition includes separation of functional concerns, temporal or sequencing concerns, and organizational concerns.		

Category: Method Description Desired/Optional Characteristics

#	Item	Rating -3 to +3	Confidence L, M, or H
28.	Provides guidance for verification.		
29.	Provides examples that run through complete execution of the method.		
30.	Provides instructions on human engineering, dependability engineering, and performance engineering.		
31.	Provides algorithmic steps for execution of the method.		
32.	Provides instructions for validation.		
33.	While exploiting the features of organization's standard programming language(s), presents alternatives for other languages wherever possible.		
34.	Provides reusable issues, alternatives, rationales, and decisions.		
35.	Provides a graphical description of integration of subcomponents within the method and integration of the method within the process(es).		
36.	Provides references that should be followed, show evidence for maturity, explain prior work, describe rejected alternatives.		

Category: Kinds of Engineering Knowledge

#	Item	Rating -3 to +3	Confidence L, M, or H
37.	Fundamental design concepts		
38.	Criteria and specifications		
39.	Theoretical tools		
40.	Quantitative data		
41.	Practical considerations		
42.	Know how: procedures, ways of thinking, judgmental skills, design instrumentalities		
43.	Normative: standard, quantitative solutions based on subjective value judgements		

Category: Kinds of Engineering Knowledge (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
44.	Rational: quantitative analysis and algorithms that tell how to find solution		
45.	Argumentative: broad participation requirements or process for interested parties		
46.	Heuristic: common sense rules derived from experience and judgement		

Category: Methods Conformance

#	Item	Rating -3 to +3	Confidence L, M, or H
47.	Consideration will be given for each method to cover the standard life-cycle models.		
48.	Each method conforms to the process architecture.		
49.	Each method conforms to the central glossary.		
50.	Each method conforms to standard formats.		
51.	Each method conforms to a standard process notation.		

Category: Level of Detail

#	Item	Rating -3 to +3	Confidence L, M, or H
52.	The contents reflect a level of detail appropriate to the purpose of the document.		
53.	Adequate elaboration is provided in each area.		
54.	Input is described in sufficient detail.		

Category: Compatibility/Interoperability

#	Item	Rating -3 to +3	Confidence L, M, or H
55.	Product addresses use on standard platforms.		
56.	Product addresses use with standard interfaces.		

Category: Level of Integration

#	Item	Rating -3 to +3	Confidence L, M, or H
57.	Product is internally integrated.		
58.	Product is integrated with rest of software engineering environment and practices.		
59.	Addresses common data integration.		
60.	Addresses common conceptual bases integration.		
61.	Addresses common control integration.		
62.	Addresses common presentation.		
63.	Addresses common process or procedures.		
64.	Addresses tool interoperability.		

Category: Planned Improvements

#	Item	Rating -3 to +3	Confidence L, M, or H
65.	Improvements provide gateway to other innovations.		
66.	Consideration given to cost of conversion to future replacement.		
67.	Improvements to definition and supporting materials made with ease of usability in mind for customers (especially instantiators and users).		

Category: Method Automation

#	Item	Rating -3 to +3	Confidence L, M, or H
68.	Descriptions of how to use supporting tools or procedures are available.		
69.	Descriptions of the associated input, outputs, and data for a supporting tool are defined.		
70.	Consideration was given to the cost-effectiveness of available, supporting tools or of the construction of new, supporting tools.		
71.	The resulting software products was verified against the tool/procedure definition for the method.		
72.	Provides advice and suggestions for automation.		
73.	Provides precise instructions on automation.		
74.	Status messages, error messages, and recovery are explained.		

Category: Scope and Availability of Supporting Items

#	Item	Rating -3 to +3	Confidence L, M, or H
75.	Maintenance support available		
76.	Maintenance aids available		
77.	Training available		
78.	Consulting available		
79.	Pilot projects available		
80.	User groups available		
81.	User aids		
82.	Operation aids		
83.	Technology insertion aids Includes information on how to tailor product to organization, success stories, and codevelopment of methods with member companies.		
84.	Other support (e.g., hotline)		
85.	Reliability The ability to perform the promised service dependably and accurately.		
86.	Responsiveness The willingness to help customers and to provide prompt service.		
87.	Assurance The knowledge and courtesy of employees and their ability to convey trust and confidence.		

Category: Style and Aesthetics

#	Item	Rating -3 to +3	Confidence L, M, or H
88.	Definition has style Includes harmonious, balanced, elegant, unified, complete, refined, fluent, clear.		
89.	Definition is aesthetic Includes attractive, interesting to view.		
90.	Demonstrates technical skill in content and communication.		
91.	Has perceived quality.		

Category: Style and Aesthetics (cont.)

#	Item	Rating -3 to +3	Confidence L, M, or H
92.	Is well-crafted, well-made.		
93.	Is expressive.		
94.	Is interesting to study and/or use.		

Category: Role Usage Summary

#	Item	Rating -3 to +3	Confidence L, M, or H
95.	Contains information required by evaluators and adopters checklist under role usage.		
96.	Contains information required by Instantiators checklist under role usage.		

B.2.6 COMPETITION*Category:* Benchmark Information

#	Item	-3 to +3	L, M, or H
1.	Has overall relative advantage Over professional books, government-funded definitions and your organization definitions.		
2.	Positioning of the product and compared products Positioning is clearly stated and consistently reflected throughout the definition and supporting products and services.		
3.	Product uniqueness For example, product has unique features for customer, did a unique task for the customer, and/or was a highly innovative product that was new to the market.		
4.	Superiority For example, product is superior to competing products in meeting customer's needs and/or the product is higher quality than the competitors.		
5.	Novelty Product has originality (novel, unusual, unique, original and/or ingenious), is germinal (trend setting, influential, revolutionary, and/or radical), and/or is transformational (has an impact on society or culture).		
6.	Usability Product has less prerequisites, has lower level prerequisites, takes less time to learn, takes less time to perform, causes fewer mistakes and problems, takes less time to modify products, is preferable, is less frustrating to use, would be used again, and would be recommended to colleagues.		
7.	Market and business advantages Marketing and business factors include appearance, conspicuousness, market differentiation, social value, service availability, and pricing competitiveness.		

Category: Definition Competition Information

#	Item	-3 to +3	L, M, or H
8.	Definition includes information that was gathered on competition and alternative methods through such techniques as benchmarking and literature searches.		

This page intentionally left blank.

LIST OF ABBREVIATIONS AND ACRONYMS

CASE	computer-aided software engineering
CM	configuration management
CMM	Capability Maturity Model
DOD	Department of Defense
ESP	Evolutionary Spiral Process
ETVX	Entry-Task-Validation-eXit
FAA	Federal Aviation Administration
IDEF	Integrated DEFinition
IE	Improvement Efforts
ISO	International Organization of Standardization
MPDM	Managed Process Definition Methodology
NASA	National Aeronautics and Space Administration
NSA	National Security Agency
OPD	Organizational Process Development
PAD	Project Application Development
PASTA	Process and Artifact State Transition Abstraction
PAT	Process Action Team
PERT	Program Evaluation and Review Technique
PLD	Product-Line-Based Product and Process Development
PPA	Program Process Architecture
QA	quality assurance
R & D	research and development

RFP	Request for Proposal
RIN	Role Interaction Nets
SADT	Structured Analysis and Design Technique
SDT	State Transition Diagram
SEI	Software Engineering Institute
SEPG	Software Engineering Process Group
SPICE	Software Process Improvement Capability dEtermination
TQM	Total Quality Management
V & V	verification and validation
VCOE	Virginia Center of Excellence for Software Reuse and Technology Transfer

GLOSSARY

Class template	Class templates are derived from meta-class templates and add information unique to that template.
Constraint	Process constraints describe the limiting conditions associated with the activation, performance, or cessation of an event. Whereas supports can be viewed as those things required to enable or make the right things happen, constraints can be viewed as those things required to disable or prevent the wrong things from happening. In this guidebook, constraints have been divided into two general types: internal constraints and external constraints.
Contributing template	These are any non-leaf-node templates.
Cycle	A traversal of all five sectors of the spiral model, which donates that some aspect of the product has matured by a specific amount.
End-product	The product that results from a process definition and modeling effort. Examples include guidebooks, training material, subsections of proposals, operations manuals, project plans, etc.
Estimate of the situation (EoS)	A document that identifies the project's goals, strategies, product and process assumptions, and the assets available for performing a project.
External constraints	External constraints include all factors that may limit or constrain how an activity proceeds, that are not directly attributable to local authority (which are modeled as internal constraints). Examples of external influences that may constrain an activity include quality requirements, corporate standards, division policies, engineering procedures, process guidelines, and management directives. External constraints differ from internal constraints in that they are typically not subject to discretionary use—they are intended to be, and expected to be, explicitly followed, regardless of project-specific issues.

Final template	These are the leaf-node templates.
Foundation template	All templates are derived from a common <i>foundation template</i> .
Guidance	The use of a process definition (constraint) by an observer or process agent to provide the enacting process agent with the legal set of process step options at any point of the enactment of the observed process. This may involve process cues, process interaction, or process management.
Internal constraints	Internal process constraints are typically managerial in nature and usually take the form of authority and permission. Examples of internal constraints include management authority or permission required before an event can commence. Internal constraints also convey authority to roles to suspend events, cancel activities, recommence activities, cease activity, etc. In all cases, internal constraints are always coupled with a role (typically a role signifying lead or managerial responsibility, but in all cases a role signifying—by definition—some form of authority). As a rule, internal constraints are those constraints that you have authority to change, countermand, enforce, etc.
Meta-class template	Meta-class templates inherit common fields from the foundation template.
Policy	A guiding principle; a process constraint, usually at a high level, that focuses on certain aspects of a process and influences the enactment of that process.
Precision	The degree to which the process definition completely specifies all the actions needed to produce accurate results. That is, a precisely defined process, executed with fidelity, produces an accurate result.
Predictability	An indication that either the process is intended to terminate and does terminate or that the process is intended to be nonstop and that it does continue until terminated by a control process (or its agent).
Process	A series of actions or operations conducting to an end. A series of actions intended to reach a goal, possibly resulting in products.

Process architecture	<p>A conceptual framework for incorporating process elements in consistent ways (or for signaling that the process element is incompatible with the architecture).</p> <p>A framework within which project-specific processes are defined.</p>
Process control	<p>The external influence over process enactment by other enacting processes. This influence may be driven by process evaluation and may be through control of the process enactment state, reassignment of resources, or change of process goals through process evolution.</p>
Process definition	<p>An instantiation of a process design for a specific project team or individual. It consists of a partially ordered set of process steps that is enactable. Each process step may be further refined into more detailed process steps. A process definition may consist of (sub)process definitions that can be concurrently enacted. Process definitions, when enactable by humans, are referred to as process scripts. Process definitions for nonhuman enactment are referred to as process programs.</p>
Process evolution	<p>The evolution of process definitions (static) as well as the evolution of enacting processes (dynamic), e.g., nonstop processes. Both static and dynamic change must be managed to ensure stability of the process and control over the process results.</p>
Process model	<p>A possibly partial process definition for the purpose of modeling certain characteristics of an actual process. Process models can be analyzed, validated, and, if enactable, simulates the modeled process. Process models may model process architecture, design, project plans, etc. Process models are at times used to predict process behavior.</p>
Process modeling	<p>Process modeling both extends and constrains process definition by requiring that the process model adheres to a predefined set of objects, relationships, methods, and structural conventions, the latter of which are often rendered graphically.</p>
Process representation	<p>A general term referring to the combined or sequential efforts of jointly performing process definition and process modeling.</p>

Process supports

A process support is any non-throughput item that is needed by an activity for the activity to be performed. Activities need products, as that typically is the purpose of activities: to accept one or more products, modify, manipulate, inspect, and possibly create one or more new products, and pass those along to other activities. However, more is needed by an activity than just the products. These nonproduct items are all modeled as supports. Two common types of support include roles and resources.

Products

Products represent the vast majority of artifacts that pass through a process. Examples include code modules, end-user guidebooks, circuit boards, and anything else tangibly produced by a process. Products can be decomposed into subproducts, sub-subproducts, etc.

Project

An enactable or enacting process whose architecture has control processes (project management) and enacting processes performing the project tasks.

Project management

An enactable or enacting process whose goal is to create project plans and, when authorized, instantiate them, monitor them, and control their enactment. These responsibilities are commonly known as project planning (i.e., development of process plans) and project control (i.e., process evaluation of plan information and process control to make adjustments, if necessary).

Project manager

A human agent enacting the control process responsible for the execution of a project.

Redundancy

A process task or step that is not required by an error-free enactment. Redundancy thus compensates for human or other errors in process enactment.

Research	Research is a by-product of a process; but it differs from products in that research is considered intangible. If for instance, the research leads to a technical paper, that technical paper is modeled as a product. However, if experiments or investigations are being performed within one or more events, but nothing tangible is available as evidence of the work, the throughput can still be explicitly modeled as a research (intangible) artifact. As with products, research can be decomposed into subresearch, sub-subresearch, etc. This decomposition is captured within a model by the inclusion relation.
Resources	Resources are nonhuman items needed to support an event. Examples include equipment, office space, supplies, funding, etc. All items that might be required to support an event can be modeled as resources. Resources can be decomposed (using the inclusion relation) so that while one level of event abstraction shows that the training building is required, at a lower or more detailed level of abstraction the support might show that only a small classroom is actually required.
Robustness	The degree to which the process rejects unauthorized process control and/or modification (intrusion).
Role	Roles commonly represent either individual humans or humans working in concert toward a common goal or set of goals. Consequently, "programmer," "manager," "clerk," etc., all define roles that can be assumed by individuals. However, "programming team," "inspection department," and "quality assurance division" also define roles. In the latter case, the roles are essentially organizational roles as opposed to individual. For process definition, roles can be defined at all levels of abstractions.
Spiral	One or more cycles.
Subclass template	Subclass templates are used to further refine and distinguish process information.
Task	A process (step), typically enacted by a human, requiring process planning and control.

Tiers of Usage

Tiers of usage separate process definition work by degree of formality. Tier 1 usage requires the least degree of formality. Tier 4 usage requires a high degree of formality.

REFERENCES

- 1986 A Spiral Model of Software Development and Enhancement.
ACM Software Engineering Notes 11:22-42.
- 1988 A Spiral Model of Software Development and Enhancement.
IEEE Computer 21:61-72.
- Boehm, B., and F. Belz
1989 "Experiences With the Spiral Model as a Process Model
Generator," In *Proceedings of the 5th International Software
Process Workshop*, pp. 43-45.
- 1992 "CASE & the Management of Risk." Presented at the CASE
World Conference, Santa Monica, California, 18-20 February.
- Coleman, Glenn L.,
Charles P. Ellison,
Gentry P. Gardner,
Daniel L. Sandini, and
John W. Brackett
1990 *Experience in Modeling a Concurrent Software System Using
STATEMATE*. Proceedings of the 1990 IEEE International
Conference on Computer Systems and Software Engineering,
pp. 104-108.
- Department of Defense
1985 *Technical Reviews and Audits for Systems, Requirements, and
Computer Programs*, DOD-STD-1521B. Washington, D.C.:
Department of Defense.
- 1988 *Military Standard: Defense System Software Development*,
DOD-STD-2167A. Washington, D.C.: Department of Defense.
- Fagan, M.E.
1986 "Advances in Software Inspections." In *IEEE Transactions on
Software Engineering*, Volume SE-12, Number 7, pp. 744-751.
- Feiler, Peter H., and
Watts S. Humphrey
1992 *Software Process Development and Enactment: Concepts and
Definitions*, CMU/SEI-02-TR-4. Pittsburgh, Pennsylvania:
Software Engineering Institute, Carnegie Mellon University.
- Grove, Andrew S.
1983 *High Output Management*. New York, New York: Random
House.
- Harel, David
1988 *Statecharts: A Visual Formalism for Complex Systems*.
Department of Applied Mathematics, The Weizmann Institute
of Science, Rehovot, Israel. 1986. (Eventually published under
the same title in 1987 in *Science of Computer Programming*,
8(1987):231-274.)

- Henderson, W., and P. Taylor
1991
Embedded Processes in Stochastic Petri Nets. *IEEE Transactions on Software Engineering* 17, 2.
- Implementation Management
Associates
1992
Accelerating Change Workshop. Brighton, Colorado: IMA, Inc.
- Kellner, M.
1989
Representation Formalisms for Software Process Modeling. *Proc. 4th International Software Process Workshop*.
- Kolman, Bernard, and
Robert C. Busby
1984
Discrete Mathematical Structures for Computer Science. Englewood Cliffs, New Jersey: Prentice-Hall Inc.
- Levis, Alexander
1992
Class lecture notes during ECE590/SYST659 Spring 1992 course at George Mason University.
- Marca, David A., and
Clement L. McGowan
1988
SADT: Structured Analysis Design Technique. McGraw-Hill Book Company.
- Paulk, Mark, William Curtis,
Mary Beth Chrissis, and
Charles V. Weber
1993
Capability Maturity Model for Software, version 1.1 CMU/SEI-93-TR-24. Pittsburgh, Pennsylvania: Software Engineering Institute.
- Radice, Ronald A., and
Richard W. Phillips
1988
Software Engineering: An Industrial Approach. Englewood Cliffs, New Jersey: Prentice-Hall Inc.
- Redwine, S.T.
1991
"Process Architecture Issues," In *Proceedings of the Seventh International Software Process Workshop*, Yountville, California, pp. 117-120.
- Redwine, S.T. and W.E. Riddle
1985
"Software Technology Maturation." In *Proceedings 8th International Conference on Software Engineering*, IEEE.
- Sage, Andrew
1993
"Systems Engineering for Software Intensive Systems." Presentation by Software Productivity Consortium, September 23.
- Sanden, Bo
1992a
Software Systems Construction: Sequential and Concurrent Designs Implemented in Ada. Pre-Published Textbook used at George Mason University for Spring 1992. Material is the property of Bo Sanden and Prentice Hall.
- 1992b
3.0 Statecharts. Supplemental Handout #2 for INFT821. Spring 1992: George Mason University.

- Singh, Baldev
1992
Interaction Roles: A Coordination Model, CT-084-92. MCC Technical Report.
- Singh, Baldev, and
Gail L. Rein
1992
Role Interaction Nets (RINs): A Process Description Formalism, CT-083-92. MCC Technical Report.
- Software Productivity
Consortium
1992
Process Definition and Modeling Guidebook, SPC-92041-CMC, version 01.00.02. Herndon, Virginia: Software Productivity Consortium.
- 1993a
Managing Process Improvement: A Guidebook for Implementing Change, SPC-93105-CMC, version 01.00.06. Herndon, Virginia: Software Productivity Consortium.
- 1993b
Using New Technologies: A Technology Transfer Guidebook, SPC-92046-CMC, version 02.00.08. Herndon, Virginia: Software Productivity Consortium.
- 1994
Process Engineering With the Evolutionary Spiral Process Model, SPC-93089-CMC, version 01.00.06. Herndon, Virginia: Software Productivity Consortium.
- Venkatraman, N.
1991
"IT-Induced Business Reconfiguration." *The Corporation for the 1990's*. Edited by Michael S. Scott Morton. New York: Oxford University Press, pp. 122-58.

This page intentionally left blank.

BIBLIOGRAPHY

Britton, K.H., R.A. Parker, and D.L. Parnas. "A Procedure for Designing Abstract Interfaces for Device Interface Modules," In *Proceedings, 5ICSE*. pp. 195-204, 1981.

Clements, P.C., R.A. Parker, D.L. Parnas, J.E. Shore, and K.H. Britton. *A Standard Organization for Specifying Abstract Interfaces*. NRL Report 8815, June 14, 1984.

Curtis, B. *Modeling, Measuring, & Managing Software Development Process. The M3 Life Boat for Software Tarps*. Tutorial #4, The 13th Internal Conference on Software Engineering, 1991.

Feiler, Peter, and Watts Humphrey. *Software Process Definitions Draft Document*. Pittsburgh, Pennsylvania: Software Engineering Institute, Carnegie-Mellon University, 1991.

Guindon, R. *A Framework for Building Software Development Environments: System Design as Ill-structured Problems and as an Opportunistic Process*. MCC Technical Report STP-298-88, 1988.

Kirby, J. Jr., R.C.T. Lai, and D.M. Weiss. "A Formalization of a Design Process." In *Proceedings, 1990 Pacific Northwest Software Quality Conference*. Oct. 29-31, 1990:93-114.

Osterweil, L. "Software Processes Are Software Too." In *Proceedings, 9ICSE*. March 1987.

Parnas, D.L. "On the Criteria to be Used in Decomposing a System into Modules." *Communications of the ACM* 15,12 (1972):1053-1058.

Parnas, D.L., and P.C. Clements. "A Rational Design Process: How and Why to Fake It." *IEEE Transactions on Software Engineering* (February 1986).

Potts, C. "A Generic Model for Representing Design Methods." In *Proceedings, 11ICSE*, 1989.

Potts, C., and G. Bruns. "Recording the Reasons for Design Decisions." In *Proceedings, 10ICSE*, April 1988.

Rendes, Barry, and Ralph M. Stair, Jr. *Quantitative Analysis for Management*. Third Edition. Allyn and Bacon Inc., 1988.

This page intentionally left blank.