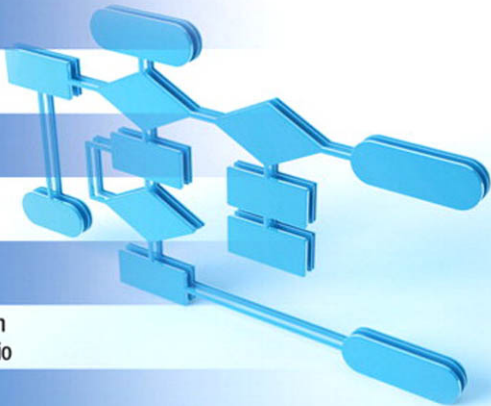


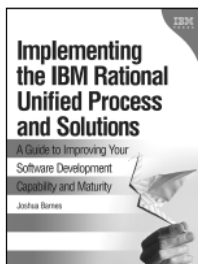
# Work Item Management with IBM Rational ClearQuest and Jazz

A Customization Guide



Shmuel Bashan  
David E. Bellagio

# Related Books of Interest

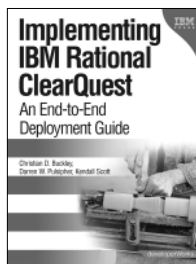


## Implementing the IBM® Rational Unified Process® and Solutions

By Joshua Barnes

ISBN-10: 0-321-36945-9

This book delivers all the knowledge and insight you need to succeed with the IBM Rational Unified Process and Solutions. Joshua Barnes presents a start-to-finish, best-practice roadmap to the complete implementation cycle of IBM RUP—from projecting ROI and making the business case through piloting, implementation, mentoring, and beyond. Drawing on his extensive experience leading large-scale IBM RUP implementations and working with some of the industry's most recognized thought leaders in the Software Engineering Process world, Barnes brings together comprehensive “lessons learned” from both successful and failed projects. You'll learn from real-world case studies, including actual project artifacts.



## Implementing IBM® Rational® ClearQuest®

### An End-to-End Deployment Guide

By Christian D. Buckley, Darren W. Pulsipher, and Kendall Scott

ISBN-10: 0-321-33486-8

### ***Implementing IBM Rational ClearQuest***

brings together all you need to integrate ClearQuest into an over-arching change-management system that works. Drawing on decades of experience, the authors present a detailed, easy-to-use roadmap for each step of ClearQuest deployment, from evaluating business cases to planning, design, and implementation. You will find the industry's clearest, most useful explanations of ClearQuest technology here, along with real-world examples, best practices, diagrams, and actionable steps.

Sign up for the monthly IBM Press newsletter at  
[ibmpressbooks.com/newsletters](http://ibmpressbooks.com/newsletters)

# Related Books of Interest



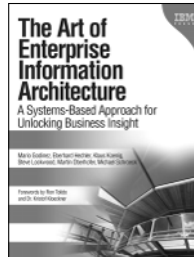
## A Practical Guide to Distributed Scrum

By Elizabeth Woodward, Steffan Surdek, and Matthew Ganis

ISBN-10: 0-13-704113-6

This is the first comprehensive, practical guide for Scrum practitioners working in large-scale distributed environments. Written by three of IBM's leading Scrum practitioners—in close collaboration with the IBM QSE Scrum Community of more than 1,000 members worldwide—this book offers specific, actionable guidance for everyone who wants to succeed with Scrum in the enterprise.

Readers will follow a journey through the lifecycle of a distributed Scrum project, from envisioning products and setting up teams to preparing for Sprint planning and running retrospectives. Using real-world examples, the book demonstrates how to apply key Scrum practices, such as look-ahead planning in geographically distributed environments. Readers will also gain valuable new insights into the agile management of complex problem and technical domains.



## The Art of Enterprise Information Architecture

### A Systems-Based Approach for Unlocking Business Insight

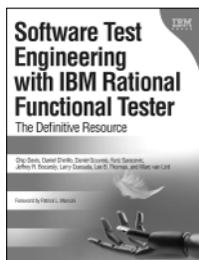
By Mario Godinez, Eberhard Hechler, Klaus Koenig, Steve Lockwood, Martin Oberhofer, and Michael Schroeck

ISBN-10: 0-13-703571-3

Tomorrow's winning "Intelligent Enterprises" will bring together far more diverse sources of data, analyze it in more powerful ways, and deliver immediate insight to decision-makers throughout the organization. Today, however, most companies fail to apply the information they already have, while struggling with the complexity and costs of their existing information environments.

In this book, a team of IBM's leading information management experts guide you on a journey that will take you from where you are today toward becoming an "Intelligent Enterprise." Drawing on their extensive experience working with enterprise clients, the authors present a new, information-centric approach to architecture and powerful new models that will benefit any organization.

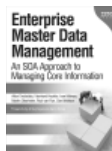
# Related Books of Interest



## Software Test Engineering with IBM Rational Functional Tester The Definitive Resource

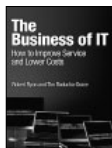
By Chip Davis, Daniel Chirillo, Daniel Gouveia, Fariz Saracevic, Jeffrey B. Bocarsley, Larry Quesada, Lee B. Thomas, and Marc van Lint  
ISBN-10: 0-13-700066-9

If you're among the thousands of developers using IBM Rational Functional Tester (RFT), this book brings together all the insight, examples, and real-world solutions you need to succeed. Eight leading IBM testing experts thoroughly introduce this state-of-the-art product, covering issues ranging from building test environments through executing the most complex and powerful tests. Drawing on decades of experience with IBM Rational testing products, they address both technical and nontechnical challenges and present everything from best practices to reusable code.



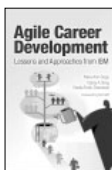
## Enterprise Master Data Management

An SOA Approach to Managing Core Information  
Dreibelbis, Hechler, Milman, Oberhofer, van Run, Wolfson  
ISBN-10: 0-13-236625-8



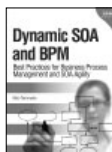
## The Business of IT

How to Improve Service and Lower Costs  
Robert Ryan, Tim Raducha-Grace  
ISBN-10: 0-13-700061-8



## Agile Career Development

Lessons and Approaches from IBM  
Mary Ann Bopp, Diana A. Bing, Sheila Forte-Trammell  
ISBN-10: 0-13-715364-3



## Dynamic SOA and BPM

Best Practices for Business Process Management and SOA Agility  
Marc Fiammante  
ISBN-10: 0-13-701891-6



## Multisite Commerce

Proven Principles for Overcoming the Business, Organizational, and Technical Challenges  
Lev Mirlas  
ISBN-10: 0-13-714887-9

Sign up for the monthly IBM Press newsletter at [ibmpressbooks.com/newsletters](http://ibmpressbooks.com/newsletters)



## **Praise for *Work Item Management with IBM Rational ClearQuest and Jazz***

“Dave and Shmuel have mastered both CQ ALM and Jazz and produced a primer introducing these to the IBM Rational Tools audience. This is a great starting point for implementing the integration.”

—*Robert W. Myers*  
*CQ ALM Architect*  
*IBM Rational*

“This book is an excellent introduction to how to think about workflows, a topic that has been severely lacking in discussions of implementing tools that govern workflows. Without this understanding of workflows, good tools are often poorly implemented and therefore don’t yield the ROI expected. Although this book uses Rational tools to describe implementation fine points, the information presented will be very useful for doing the groundwork for implementing any workflow-supporting tools.”

—*Chuck Walrad*  
*Managing Director*  
*Davenport Consulting*

“Dave Bellagio and Shmuel Bashan provide thorough and practical coverage of how to implement a work item change management process using IBM ClearQuest or the Jazz Platform. A must-read for professionals involved in implementing a change management process with IBM ClearQuest or with one of the Jazz-based products.”

—*Celso Gonzalez*  
*Coauthor of Patterns-Based Engineering: Successfully Delivering Solutions via Patterns*

“Application Lifecycle Management (ALM) is the key to success for today’s increasingly complex enterprise software delivery challenges. At the heart of ALM is work item management. This book provides an excellent review of practical approaches to work item management based on real world experience that will help you to deliver enterprise solutions more effectively. It is a great resource for the whole software delivery team.”

—*Alan W. Brown*  
*IBM Distinguished Engineer*

# **Work Item Management with IBM Rational ClearQuest and Jazz**

*This page intentionally left blank*

# **Work Item Management with IBM Rational ClearQuest and Jazz**

**A Customization Guide**

**Shmuel Bashan**

**David E. Bellagio**

IBM Press  
Pearson plc

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco  
New York • Toronto • Montreal • London • Munich • Paris • Madrid  
Capetown • Sydney • Tokyo • Singapore • Mexico City

[ibmpressbooks.com](http://ibmpressbooks.com)

The authors and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

© Copyright 2011 by International Business Machines Corporation. All rights reserved.

Note to U.S. Government Users: Documentation related to restricted right. Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corporation.

IBM Press Program Managers: Steve Stansel, Ellice Uffer

Cover design: IBM Corporation

Associate Publisher: David Dusthimer

Marketing Manager: Stephane Nakib

Publicist: Heather Fox

Acquisitions Editor: Chris Guzikowski

Development Editors: Sheri Cain and Chris Zahn

Managing Editor: John Fuller

Designer: Alan Clements

Project Editor: Anna Popick

Copy Editor: Barbara Wood

Indexer: Jack Lewis

Compositor: The CIP Group

Proofreader: Kelli M. Brooks

Manufacturing Buyer: Dan Uhrig

Published by Pearson plc

Publishing as IBM Press

IBM Press offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales

1-800-382-3419

corpsales@pearsontechgroup.com

For sales outside the U.S., please contact:

International Sales

international@pearson.com

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both: IBM, Rational, ClearQuest, Rational Team Concert, Jazz, ClearCase, RequisitePro, BuildForge, PurifyPlus, WebSphere, DB2, DB2 Universal Database, developerWorks, Tivoli, PureCoverage, Quantify, DOORS, Lotus, and Sametime. Microsoft, Visual SourceSafe, ActiveX, Windows, VisualStudio, Access, Excel, and SharePoint are trademarks of Microsoft Corporation in the United States, other countries, or both. UNIX is a registered trademark of The Open Group in the United States and other countries. Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Other company, product, or service names may be trademarks or service marks of others.

The following terms appear throughout the book: Introduction: IBM®, Rational®, ClearQuest®, Rational Team Concert™, Jazz™, ClearCase®, RequisitePro®, Visual SourceSafe®, BuildForge®, PurifyPlus™; Chapter 1: ActiveX®, Windows®; Chapter 2: UNIX®, VisualStudio®, WebSphere®, DB2®, DB2 Universal Database™, Access®, Oracle®; Chapter 4: developerWorks®; Chapter 6: Tivoli®, Linux®, PureCoverage®, Quantify®, Excel®, Java™, DOORS®, Lotus®, Sametime®, Quickr®, SharePoint®.

*Library of Congress Cataloging-in-Publication Data*

Bashan, Shmuel, 1952-

Work item management with IBM Rational Clearquest and jazz : a customization guide / Shmuel Bashan, David E. Bellagio.

p. cm.

Includes index.

ISBN 978-0-13-700179-8 (pbk. : alk. paper)

1. Business—Computer programs. 2. Teams in the workplace--Data processing. 3. Computer software—Development—Management. 4. Jazz (Computer file) 5. Rational Clearquest. I. Bellagio, David E. II. Title

HF5548.4.M5265B37 2011

658.4'04028553—dc22

2011011419

All rights reserved. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. For information regarding permissions, write to:

Pearson Education, Inc.  
Rights and Contracts Department  
501 Boylston Street, Suite 900  
Boston, MA 02116  
Fax: (617) 671-3447

ISBN-13: 978-0-13-700179-8

ISBN-10: 0-13-700179-7

Text printed in the United States on recycled paper at RR Donnelley in Crawfordsville, Indiana.  
First printing, June 2011

*In memory of my late parents, Sari and Moni,  
for being role models of hard work, modesty, and honesty  
—Shmuel*

*To my children, Anthony, Jacob, and Mark  
—Dave*

---

# Contents

<b>Preface</b>	<b>xix</b>
<b>Acknowledgments</b>	<b>xxvii</b>
<b>About the Authors</b>	<b>xxix</b>
<b>Chapter 1 Work Items</b>	<b>1</b>
1.1 Work Item Definition	1
1.1.1 Terms Related to Work Items in ClearQuest and Jazz	1
1.2 Work Item Classification	4
1.2.1 Change Requests	4
1.2.2 Activities	5
1.2.3 Test Elements	6
1.2.4 Project-Related Work Items	6
1.3 Work Item Elements	7
1.3.1 Data	7
1.3.2 Presentation Forms	7
1.3.3 Workflow	9
1.3.4 Other Work Item Elements	11
1.4 Customization	13
1.4.1 Which Work Item Elements Can Be Customized?	14
1.4.2 Customizing Jazz Work Items	17
1.4.3 Customizing ClearQuest Record Types	18
1.5 Resources	19
1.6 Summary	20
<b>Chapter 2 Disciplines: Requirements, Analysis &amp; Design</b>	<b>21</b>
2.1 Requirements	23
2.1.1 Gathering Requirements	23
2.1.2 Defining and Documenting Requirements	25
2.1.3 Getting Agreement	28
2.1.4 Using Agile Practice	28
2.1.5 Maintaining Requirements	28



2.2 Analysis & Design	29
2.2.1 Defining the Types of Clients	29
2.2.2 Defining the Infrastructure Architecture	30
2.2.3 Choosing a Database	32
2.2.4 ClearQuest Schema High-Level Design	33
2.2.5 Defining the Data Fields	35
2.2.6 Defining the Workflow	35
2.2.7 Designing the User Interface (Forms)	36
2.3 Design Patterns	36
2.3.1 Closing Pattern	37
2.3.2 Triage Pattern	37
2.3.3 Parent Control Pattern	38
2.3.4 Child Control Pattern	38
2.3.5 Dead End Pattern	39
2.3.6 Data Hierarchy Pattern	39
2.3.7 Superuser Modification Pattern	39
2.3.8 Resolution Pattern	40
2.4 Review and Sign Off Design Models	40
2.5 Resources	40
2.6 Summary	40
<b>Chapter 3 The Workflow</b>	<b>43</b>
3.1 Software Development Processes	44
3.1.1 The Rational Unified Process (RUP) Method	44
3.1.2 OpenUP Method	45
3.2 Process Representation	45
3.2.1 Creating the State Transition Matrix	49
3.3 The States	50
3.3.1 Basic Stages in Workflow	50
3.3.2 State Types	53
3.4 Dynamic Workflow	56
3.4.1 Background	56
3.4.2 The Technique	56
3.4.3 Automatically Move to Another State	57
3.4.4 One Record Type Having Several State Machines for Each Issue Type	59
3.4.5 A State Machine for Each Issue	61
3.5 ClearQuest ALM Schema Workflow	65
3.6 Jazz Workflow	66
3.7 Subflow	68
3.7.1 More Information	69
3.7.2 Build Approval	76
3.8 Summary	78

<b>Chapter 4</b>	<b>The Data</b>	<b>81</b>
4.1	Work Item Content	83
4.1.1	Work Item Description	83
4.1.2	Location	86
4.1.3	Environment	87
4.1.4	Internal Impacts	88
4.1.5	External Impacts	89
4.1.6	Corrective Actions	91
4.1.7	Times	92
4.1.8	Tests	95
4.1.9	History	96
4.1.10	Additional Data	98
4.1.11	Quality Assurance	103
4.2	State-Based Objects	104
4.3	Stateless Objects	104
4.4	Object Relations	106
4.4.1	ClearQuest Single Relationship	106
4.4.2	ClearQuest Multiple Relationship	108
4.4.3	Back Reference	110
4.4.4	More on ClearQuest Unique Key	113
4.4.5	Jazz Links	115
4.5	Data Representation	117
4.5.1	ClearQuest Data Representation	117
4.5.2	Jazz Work Items Data Representation	118
4.6	ClearQuest Scripts	120
4.6.1	HasAttachment	120
4.6.2	Limit Attachment Size (Perl)	121
4.6.3	Convert Full_Name to Login_Name	122
4.6.4	Create Parent from Child	123
4.7	Summary	126
<b>Chapter 5</b>	<b>Roles</b>	<b>127</b>
5.1	Understanding Typical Problems	128
5.1.1	Enforcing Security Control	128
5.1.2	Unassigned Change	128
5.1.3	Assigning the Wrong Person	128
5.1.4	Blocking Assignment	129
5.2	Understanding Terms and Concepts	129
5.2.1	Basic Definitions	130
5.2.2	Role Type (Cardinality)	130
5.2.3	Roles, Areas, and Groups	131
5.2.4	Key Roles in the Change Request Process	131
5.3	Possible Solutions	132
5.3.1	Implementing Roles with ClearQuest Groups	132

5.3.2	Implementing Roles Implicitly	133
5.3.3	Using Roles Stateless Record Type (with Static Roles)	136
5.3.4	User-Defined Roles	137
5.4	Security and Roles	139
5.5	Roles in the ClearQuest ALM Schema	140
5.6	Roles in Jazz	142
5.7	Code Examples	144
5.7.1	SQL Command for Group Method	144
5.7.2	Hook to Automatically Set Responsible Based on Role of Type Single	145
5.7.3	Hook to Automatically Set Choices Based on Role of Type Multiple	146
5.7.4	Hook to Automatically Set Responsible Based on Role Object	147
5.7.5	Hooks for User-Defined Roles	150
5.8	Summary	151
<b>Chapter 6</b>	<b>Integrations</b>	<b>153</b>
6.1	Introduction	153
6.2	ClearQuest Integrations	155
6.2.1	ClearQuest Packaged (Built-in) Integrations	156
6.2.2	Creating New Integrations with ClearQuest	162
6.2.3	Third-Party Integrations to ClearQuest	176
6.3	Jazz Products Integrations	177
6.3.1	Rational Quality Manager Integrations	177
6.3.2	Rational Team Concert Integrations	187
6.3.3	Building a New Jazz Integration	192
6.4	Resources	195
6.4.1	ALM	195
6.4.2	Jazz	195
6.4.3	ClearQuest	195
6.4.4	General Information	196
6.5	Summary	196
<b>Chapter 7</b>	<b>Disciplines, Part 2</b>	<b>197</b>
7.1	Implementation Discipline	197
7.1.1	ClearQuest Implementation Tasks	198
7.1.2	Jazz Implementation Tasks	199
7.2	Testing Discipline	199
7.3	Deployment Discipline	201
7.3.1	Preparation	201
7.3.2	Installations	202
7.3.3	Setting Up the Environment	202
7.3.4	Deploying Customizations	203
7.3.5	Importing Initial Data	204
7.3.6	Training	206
7.3.7	Following Up on System Adoption	207

7.4 Maintenance	207
7.4.1 Defining the Change Process	207
7.4.2 Ongoing Support	208
7.4.3 Improving Maintainability	208
7.5 ClearQuest Tool Mentor	208
7.5.1 Importing Records with References	208
7.5.2 Importing Updates	210
7.5.3 Creating a Test Environment	212
7.6 Jazz Tool Mentor	217
7.6.1 Creating a Jazz Project with the Common Process Template	217
7.7 Resources	220
7.8 Summary	220

<b>Chapter 8 Development</b>	<b>221</b>
8.1 ClearQuest Schema Development	222
8.1.1 Common Schema (ALM)	222
8.1.2 Implementing Patterns	222
8.1.3 Employing Reusable Assets	234
8.1.4 Using ClearQuest Packages	238
8.1.5 Understanding Session Variables	239
8.2 Parallel Development	240
8.2.1 Coding Hooks	241
8.2.2 Record Types	241
8.2.3 Designing Forms and Tabs	243
8.3 Comparing and Merging Schema Versions	245
8.4 Storing Hooks Externally	249
8.5 Releasing a Version to Production	250
8.5.1 Developer Testing	250
8.5.2 System Testing	252
8.5.3 Promotion to Production	252
8.6 Globally Distributed Development (GDD) Considerations and ClearQuest MultiSite (CQMS)	253
8.6.1 Upgrading the Schema	253
8.6.2 Addressing Mastership Changes	253
8.6.3 Testing the Mastership	254
8.7 ClearQuest Script Debugging	255
8.7.1 Employing the <code>MsgBox()</code> Function	255
8.7.2 Employing the <code>OutputDebugString()</code> Method	256
8.7.3 Debugging with Tracing Information	258
8.8 Other Development Considerations	260
8.8.1 Choosing a Scripting Language	261
8.8.2 When Is a Stateless Record Type Required?	261
8.8.3 Dealing with Records That Have More Than One Field as Unique Key	261
8.8.4 Organizing Global Scripts	262

8.8.5	Devising a Naming Convention	262
8.8.6	Storing the old_id Field for Future Import	264
8.8.7	Dealing with Long Selection Lists	265
8.8.8	Updating a Dynamic List	271
8.8.9	Using Hard-Coded Data	272
8.9	Web Considerations	274
8.9.1	Enable Button Hooks	274
8.9.2	Field Dependency	275
8.9.3	Other Limitations	276
8.10	Preparing for the Future	277
8.11	Resources	277
8.11.1	ClearQuest	277
8.11.2	Jazz	277
8.12	Summary	278
<b>Chapter 9</b>	<b>Metrics and Governance</b>	<b>279</b>
9.1	Metrics	279
9.1.1	Types of Metrics	280
9.1.2	Metrics Strategy	280
9.1.3	Supporting Data for Metrics	284
9.1.4	Tools	284
9.2	Governance	287
9.2.1	Process Control and Automation	288
9.2.2	Permissions (Access Control and Security)	288
9.2.3	Monitoring	289
9.2.4	Governance with ClearQuest	290
9.2.5	Governance with the ClearQuest ALM Schema	297
9.2.6	Governance with Rational Team Concert	299
9.3	Resources	301
9.3.1	Metrics and Governance	301
9.3.2	Jazz Reports	302
9.3.3	Data Warehouse	303
9.3.4	BIRT Reports	304
9.3.5	C/ALM Reports	304
9.4	Summary	305
<b>Chapter 10</b>	<b>Test Management and Work Items</b>	<b>307</b>
10.1	What Is Rational Quality Manager?	307
10.2	Understanding Test Entities and Work Items	307
10.3	Work Items in the Test Process	310
10.4	Customization	316
10.4.1	Customizing Jazz Work Items	317
10.4.2	Testing Specific Work Items	318
10.5	Summary	324

<b>Chapter 11</b>	<b>Managing Agile Projects</b>	<b>325</b>
11.1	Defining Agile Development	325
11.2	Agile and Scrum in a Nutshell	326
11.3	Realization with Rational Team Concert	330
11.4	Realization with ClearQuest	337
11.4.1	Required Data	339
11.4.2	Understanding the Workflows of Each Record Type	343
11.4.3	Understanding Metrics in Agile Development	345
11.5	Agile with the ALM Schema	346
11.6	Resources	350
11.6.1	Materials by Scott Ambler	350
11.6.2	DeveloperWorks Articles	350
11.6.3	Other Information	350
11.7	Summary	351
<b>Chapter 12</b>	<b>Sample Applications and Solutions</b>	<b>353</b>
12.1	Collaborative ALM with Jazz-Based Tools	353
12.1.1	Jazz C/ALM	354
12.2	User-Defined Fields in ClearQuest	356
12.2.1	Defining Choice Lists	358
12.2.2	Defining Requiredness	360
12.3	Service Level Agreements (SLAs) in ClearQuest	363
12.3.1	Background	363
12.3.2	The Topic	364
12.3.3	SLA Definition	364
12.3.4	Activate the SLA Rules	365
12.3.5	Notifications	365
12.3.6	Providing Governance Reports	366
12.3.7	The External Program	366
12.4	ClearCase, ClearQuest ALM, Build Forge Integrated Solution Architecture	367
12.4.1	Understanding the Work Projects	368
12.4.2	Developers Work on Activities within a Project	368
12.4.3	Continuously Validating through a Build and Validation Process	369
12.4.4	Creating the Task to Integrate the Baseline That Includes a Defect Fix	370
12.4.5	Releasing Periodically through Stable Composite Baselines	371
12.4.6	Working on Test Artifacts	372
12.5	Manage Release Promotion	374
12.5.1	Current Status Assessment	374
12.5.2	Solution	375
12.5.3	Process Components	376
12.5.4	The Promotion Process Model	379
12.6	Resources	382
12.6.1	Solutions Developed by Customers and Rational Staff	382

12.6.2 SLA	383
12.6.3 ALM and C/ALM	383
12.6.4 Application Lifecycle Management with Rational ClearQuest	383
12.7 Summary	384
<b>Index</b>	<b>387</b>

---

# Preface

Almost everyone has had the need at one point or another to keep a list of things that need to be done. Many people pick up a piece of paper every day and write down the things they want to attempt to accomplish that day. Such lists drive many people's lives. Some people keep them in their heads, but as we get older, we need to write things down more often so as not to forget what has to be done. When you are keeping track of a list of items for yourself to do, it is pretty easy to know what has been done and what has not. You only have to depend on yourself. Updating the list is easy. Having a list of the important things to be done and reviewing it many times a day might make many people more productive.

There are a few problems with creating a list of items to keep track of. One is that it gets much more complicated when many different people are working on the items. Now you have to coordinate updates to the list from all of those people. This problem becomes harder still when some of the people live in other time zones. Now it is much harder to understand who is doing what, what has been done, and, more important, what has not been done and why. So we typically create some sort of database to keep track of these things. In the simplest form, this database may be a spreadsheet. Some people start creating their own management system from scratch to deal with these lists. These systems tend to grow over time in both complexity and cost of maintenance. Pretty soon, as your needs grow, you may find that the process you created to keep track of the items does not scale, nor does it meet your needs anymore. The management system you created will also have its own list of items that need to be done, and you may find that the cost of maintaining your custom system outweighs the benefits you are receiving from it.

In the world of software and hardware development, lists of items tell us who is doing what, what has been done, what problems are being worked on, and which products are affected by these problems. These lists become the lifeblood for many individuals. Being able to accurately understand your product's status and exposure can help you make better decisions about what items should be worked on, what items you need to wait to be done, and what items are not as important as others.

The focus of this book is to help you implement solutions for dealing with many types of common patterns that crop up when managing items of work for large teams of people. In this

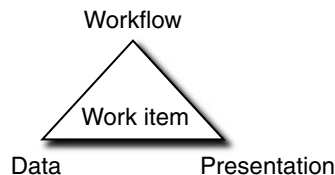


introduction we briefly explain what a work item is and the business and technical environment with which work item management is involved. We will also define and explain many basic terms that are important for you to understand as you apply the techniques provided in this book.

The book's content is organized to allow selective reading by people who are interested in only specific subjects. We explain for whom the book is intended and how different roles should read the book. Many chapters include practical sections with code examples; guidance is given for you to make the most of the provided assets so that you can reuse them in your applications. The reality is, you need some sort of management system in place to help you use the knowledge that exists in the work items. Therefore, all of the example solutions provided within the book are implemented in IBM Rational ClearQuest (CQ) and/or IBM Rational Team Concert (RTC). There are many reasons for choosing these tools to highlight the solutions we explore. Some of the reasons for using ClearQuest as the tool of choice are that it is a mature product, there are many existing examples of solutions using the tool, and its customization potential is a powerful feature. It is easier to implement the patterns we explore in ClearQuest. Theory can take you only so far; we focus on reality and the details needed to implement solutions within these two tools.

## What Is a Work Item?

A work item is an object that controls the process of performing a task. The work item contains the following elements: data, presentation forms, workflow, and possibly other elements and other objects. We call it the *work item triangle* (see Figure P.1).



**Figure P.1** Elements of the work item (the work item triangle): workflow, data, and presentation

The bottom vertices of the triangle are the data that constitutes the work item and the presentation that allows the user to view and modify the data and interact with the system. The top of the triangle is the workflow. The workflow is a series of activities performed by people having the specific roles to produce a desired outcome. In different domains the outcome is different; to achieve the outcome the three elements of the triangle must be customized so that the organization will achieve the outcome in an efficient way, with minimal risk and with the highest quality.

Work items are the fundamental mechanisms for tracking and coordinating tasks within your development organization. They are governed by the workflows within your organization's process. This book will show you practical strategies for solving typical problems that will arise

when you try to implement and deploy a work item management solution based on ClearQuest or Rational Team Concert.

Chapter 1, “Work Items,” discusses work items in more detail and refers to additional materials that will help you deal with managing work items.

## The Environment

The main environment to which this book pertains is the software development environment. However, many techniques can be applied to other environments, such as hardware development.

In the software development lifecycle (SDLC) there are several phases and several disciplines, as explained in the Rational Unified Process (RUP) and demonstrated in Figure 2.1 in Chapter 2, “Disciplines: Requirements, Analysis & Design.”

It is possible that different organizational units will have ownership of the process in different phases or in different disciplines. For example, defects defined during testing may have one type of work item and defects found in development or in production may have different work item types. Another example is that the work items for project activities are different from the work items for software defect resolution. So within the software development environment there are subenvironments.

Another environment to which work item management is relevant is the systems development environment. In this environment chip, electronic device, and appliance designers and developers adopt a different development lifecycle and also use different types of work items. In some cases work items to manage software are combined with work items to manage hardware. We discuss this important subject in the book as well.

This book contains a lot of content that deals with work item customization, including detailed examples of how to customize. To meet the specific requirements of each environment it is necessary to customize the three elements of the work item, and we shall explain how to customize the data, the workflow, and the presentation using ClearQuest and Rational Team Concert.

## This Book's Content

The following sections present brief summaries of the chapters of the book.

### Chapter 1: Work Items

A work item is an object that contains the following elements: data, presentation forms, workflow, and possibly other elements and other objects.

Work items can be classified as changes (defects, enhancement requests, and features), tasks, activities, test plans, test cases, risks, builds, promotion, and others.

The chapter explains each of the elements, how they differ in types of changes, and the best practices for design and implementation (for example, when to combine defect and enhancement requests into a single element such as an issue; how to deal with both hardware and software defects).

## Chapter 2: Disciplines: Requirements, Analysis & Design

This chapter is about the best practices used to develop ClearQuest and Rational Team Concert applications. We use the parts of the RUP methodology that are suitable to these types of applications. The following disciplines are discussed:

- **Requirements:** Gather requirements from customers and stakeholders, organize, prioritize, solve conflicts, and get agreement.
- **Analysis & Design:** Define types of ClearQuest clients, define the system architecture (server configuration, network topology, and firewall), databases, schema high-level design, and user interface. We have also included a section on design patterns.

This topic is continued in Chapter 7, “Disciplines, Part 2,” where we discuss four additional disciplines.

## Chapter 3: The Workflow

In this chapter we discuss various methods of describing the workflow and propose some patterns for designing it. In addition, ClearQuest is known to have a static state machine. In this chapter you will learn an advanced technique for creating a dynamic workflow in ClearQuest. Some implementation benefits and examples are provided.

## Chapter 4: The Data

The data of work items is stored in fields of various types. For each type of work item a set of fields is required to meet the business requirements. We shall discuss what data is required for each work item and when it is required (which state in the lifecycle). We discuss classification methods, include recommendations, and give many examples. We also discuss data grouping: necessity and techniques.

The second part of the chapter explains how to make the most of the different types of fields, such as Reference, Reference\_List, Date\_Time, and others.

Performance considerations with certain types of ClearQuest hooks are explained.

## Chapter 5: Roles

A role is a key concept in RUP; we explain how to incorporate roles into your ClearQuest schema. Three techniques are explained, each one with different complexity levels and schema structures, to meet various organizational needs.

In addition, we explain how to take advantage of roles, such as how to auto-assign owners based on roles, how to populate choice lists based on roles, and how to notify people of events based on their roles.

The last section of the chapter explains roles in Jazz and in the ClearQuest Application Lifecycle Management (CQ-ALM) schema.

## Chapter 6: Integrations

The chapter starts with a brief introduction to integration types and the value of integrating applications. It is divided between ClearQuest integrations and Jazz integrations. In the ClearQuest integrations section we describe the built-in packaged integrations, that is, ClearCase, RequisitePro, Visual SourceSafe, Microsoft Project, Build Forge, Portfolio Manager, and PurifyPlus. We continue with building new integrations, explain the methods of integrating applications with ClearQuest (e-mail, import/export, API), and give some examples: expert systems, help desk, and others.

The second part of the chapter is about Jazz integrations. We describe the Jazz platform integration technology and continue with Rational Quality Manager and Rational Team Concert integrations with ClearQuest and other products.

## Chapter 7: Disciplines, Part 2

This chapter is about the best practices used to develop ClearQuest and Jazz applications. We use part of the RUP methodology to meet the needs of these types of applications. In this chapter we discuss the following disciplines:

- **Implementing:** schema development, parallel implementation
- **Testing:** building the test environment, testing methods
- **Deployment:** managing multiple environments, enabling end users in the solution
- **Maintenance:** managing change to the solution by using the solution

## Chapter 8: Development

Although developing a ClearQuest schema is in many aspects similar to code development, there are significant differences due to the special environment. In this chapter we explain the special development considerations. Some of the subjects discussed are

- Schema development tips
- Common schema
- Pattern implementation in CQ and Jazz
- Packages
- Parallel development (and multiple schemas, multiple databases)
- Versioning content
- Releasing a version to production
- Globally Distributed Development (GDD) considerations and ClearQuest MultiSite
- Preparing for future product releases

## **Chapter 9: Metrics and Governance**

The first part of the chapter is about metrics. We explain some quality metrics (such as defect density); performance metrics (how fast and efficient our process is); and how to collect, measure, and present the data. We explain the tools available to create these metrics.

The second part of the chapter is about governance. There are various aspects of governance. In this chapter we discuss the following: controls such as electronic signature, setting service level agreements (SLAs), and managing audit logs.

Another important issue that we explain is the security setting with the Security Context record and additional security measures.

## **Chapter 10: Test Management and Work items**

In this chapter we discuss work items used in the testing process. Test management requires different considerations from change management. We review the Rational Quality Manager work items and how the different types are used. Other subjects included in this chapter are the customization of work items and the customization of other test elements.

## **Chapter 11: Managing Agile Projects**

With the emerging popularity of Agile programming methods, organizations need to adapt their workflows and automation techniques. In this chapter we briefly describe some Agile methods and how ClearQuest can be used to create a workflow for those techniques. We dive into the Scrum processes and explain how to manage backlogs and sprints with ClearQuest. A ClearQuest schema is provided for the Scrum Agile method.

We discuss in detail how Scrum is realized using Rational Team Concert. We also discuss how to implement the Agile process with the CQ-ALM schema.

## **Chapter 12: Sample Applications and Solutions**

In this chapter we explain some special applications and solutions that extend existing applications.

We start with a description of a Collaborative Application Lifecycle Management integrated solution with Jazz-based products: Rational Team Concert (RTC), Rational Quality Manager (RQM), and Rational Requirements Composer (RRC).

We describe a solution to extend a ClearQuest schema with project-defined fields and an example of an SLA with ClearQuest.

The Application Lifecycle Management (ALM) solution is a good basis for many applications, and we provide some examples and techniques for how to map your solution needs to the ALM packages that come with ClearQuest. We describe an integrated solution with ClearCase, ClearQuest, and Build Forge.

Finally, we describe a solution to managing release promotion in a heterogeneous environment using ClearQuest.

## Audience for the Book

The book will appeal to many roles and to users with a wide range of interests. This book is for everyone who is interested in software change management.

The large community of ClearQuest users will find this book valuable. This includes all users involved in ClearQuest administration, all who are interested in developing new applications with ClearQuest, and those who want to integrate ClearQuest with other applications.

The growing community of Jazz customers will also find this book interesting. In addition to the theoretical parts, we provide examples of work item management within Rational Team Concert and Rational Quality Manager, and we have dedicated Chapter 10 to test management and a focus on Rational Quality Manager.

These roles within your organization will find value in this book:

- **Project managers:** This role will learn how to use work items to help manage their project's health: what metrics are important, how to triage effectively, and obtaining visibility into potentially desired workflows that meet the organization's needs. Project managers can also learn how to use ClearQuest to manage Agile projects.
- **Technical leaders:** This role will be exposed to solutions to various problems that may shed some light on a particular issue that affects the organization's current challenges, for example, how to model and track activities related to a project's development patterns using the ClearQuest ALM workflow framework.
- **SCM administrators:** This role will be exposed to strategies for implementing solutions to problems that should benefit from a complete change management solution, for example, how the ClearQuest ALM workflows integrate with ClearCase UCM stream strategies.
- **Tools engineers:** This role will be exposed to best practices and techniques for implementing solutions to common patterns that may be important to the organization's needs.
- **Test managers:** This role will find Chapter 10 of interest, especially if adopting the Jazz platform is being considered. Test managers will also be interested in various defect-tracking techniques as well as in Chapter 9.
- **QA managers:** This role can find value in Chapter 9 as well as Chapter 10, especially if adopting the Jazz platform is under consideration.
- **Process analysts:** Process control is discussed in several chapters, in the discipline chapters and in Chapters 3 and 9. Also, the examples of using ClearQuest ALM to model the development process should be of importance.
- **Experienced ClearQuest users:** These users can deepen their knowledge of change management, learn new techniques, get new ideas for improving the system they work with, and learn how to implement ideas they have.
- **Experienced RTC and RQM users:** These users can learn how to customize the Jazz work items and how to create new work item types.

## How to Read This Book

This book attempts to close the gap in the existing materials on work item management. There aren't many books or articles that discuss this subject. The book is organized in a way that will allow many users to take advantage of its contents. In each chapter we discuss the theory of the subject with examples from the industry. After discussing the theory, we dive into the practical elements of the discipline and provide implementation examples using ClearQuest, Rational Quality Manager, or Rational Team Concert. In many cases there are several solutions to the specific requirements, and we have provided proposed solutions from a lighter-weight approach to an increasingly more complex implementation.

So, how to read this book? It depends on your role and your interests.

One way is to read the book from start to finish. It is organized in a way that will make such an approach easy. For example, we have split the disciplines between two chapters in order to make sequential reading more coherent.

Another way is to first read the two chapters about disciplines: Chapter 2 and Chapter 7. They complement each other, and we have split them so that the disciplines described in each chapter are followed with the right content. If you are interested only in the theoretical part of work item management, it makes sense to read those two chapters in sequence. They include references to practical materials in other chapters so that you can learn how the discussed disciplines are implemented with the tools.

The other chapters discuss specific areas of work item management. For example, Chapter 5 discusses roles and includes implementation examples and ClearQuest scripts. If you want to improve your process governance and lifecycle efficiency, go directly to this chapter.

Another example is Chapter 11 on Agile projects. This chapter includes a theoretical part and implementation examples in ClearQuest and Rational Team Concert. If your organization is using ClearQuest and is thinking about adopting Agile practices, you must read this chapter.

You can also use it as a cookbook. If you need to resolve an issue, search in the table of contents or the index for the relevant content. You may find implementation descriptions with code examples and references to additional materials.

---

**A Note on the Code Examples** Because of page width limitations we sometimes break command lines into two or more lines and use the backslash character (\) as the break character. Also in code examples you may see some unnatural indentations for the same reason.

---

---

# Acknowledgments

This book's content is not just the effort of the authors. The material was gathered over the past ten years of using and deploying customer solutions composed of the products mentioned within this book. The authors would like to thank the host of coworkers and customers who over the years have contributed to our greater understanding of the principles of change management, the functioning of the products, and our understanding of which change management strategies work and which ones don't.

Our ever-patient editors at Pearson, Christopher Guzikowski and Raina Chrobak, deserve many thanks as well. Many thanks to our copy editor, Barbara Wood, and to our production editor, Anna Popick, for their diligence during the production process.

We would also like to thank our families for their long-suffering during the many weekends we were busy writing the book. Shmuel would like to thank his wife, Catherine, for her support and encouragement. Dave would like to thank his wife, Laura, and kids, Anthony, Jacob, and Mark, for allowing him the time to be involved in this effort.

Much of the material is directly related to the experience of the IBM Rational field teams in engagements with customers using ClearQuest. Some of the folks whom we would like to thank for their efforts in fleshing out strategies that work with many of our larger customer needs are Ariel Whol, Shai Shapira, Etan Shomrai, Alan Murphy, Allan Wagner, Daniel Diebolt, Majid Irani, Stuart Poulin, Michael Saylor, Paul Weiss, Grant Covell, Katur Patel, Marlin Deckert, David Maroshi, Bob Myers, and Raanon Reutlinger. We have probably missed someone; if so, our apologies and thanks.

Colleagues who agreed to share content deserve many thanks: Caroline Pampino for content on C/ALM; Bob Myers for content on CQ-ALM; Scott Ambler for content on Agile; David Lubanco on metrics; Sharon Weed, Bala Rajaraman, and John Wiegand for content on integrations; Patrick Streule and Nicolas Dangeville for their help with OSLC; Yuhong Yin and Steven Pitschke for assistance in building charts and content on CQ architecture; and Alan Murphy for content on CQ development and debugging.

Special thanks to those people who put in the time to review this book and provide comments to help make the information more accurate and the reading more pleasant: Scott Ambler, Bob Myers, Michael Warfield, Chuck Walrad, and Celso Gonzalez.



We have learned from every client engagement, many of them inspired with ideas and innovative solutions. Many thanks to clients who agreed to share content: Dr. Alexander Karnovsky, Gerrit Van Doorn, Joseph W. Derr Jr., and Jacob (Kobi) Welber.

Finally, our thanks go to everyone at IBM Rational Software who keeps ClearQuest and the Jazz-based product efforts moving forward. Keep up the good work.

---

# About the Authors

## Shmuel Bashan

**Shmuel Bashan** is a Senior Deployment Specialist and Mentor on IBM Rational Software's global services account team, covering European countries. Prior to this assignment, he worked as the Rational technical leader, solution architect, and Rational country leader in Israel.

Prior to joining Rational in 1997, Shmuel was a manager of a consulting firm, and prior to that a software developer of CAD/CAM applications. Shmuel holds a B.Sc. in mechanical engineering from Ben-Gurion University and an Information Systems Analyst Certificate from the Manufacturers Association of Israel.

Shmuel is an active member of the change management community and has contributed several articles and reusable assets: scripts for various solutions, advanced training materials, and workshops. He has also contributed to the RUP 8.0 Change Management practice.

Shmuel presented at several Rational user conferences in Europe and the United States:

“What's New in Automated Software Testing and ClearQuest” (Israel, 2004)

“From Requirements to Delivery” (Israel, 2005)

“Advanced Techniques in ClearQuest Customization” (Orlando, FL, 2006)

“ClearQuest Tips and Tricks” (Strasbourg, France, 2007)

“Automating Code Integration Activities with ClearQuest ALM, UCM, and Build Forge” (with David E. Bellagio, Orlando, FL, 2009)

Shmuel published the following articles in *developerWorks*:

“Adaptive Workflow in ClearQuest”

“Manage Scrum Projects with ClearQuest”

“Using Roles for Automatic Assignment in IBM Rational ClearQuest”

Shmuel resides in Israel with his beloved wife, three children, and two grandchildren. Shmuel enjoys jazz music, art cinema and theater, playing chess and bridge, jogging along the

seashore, and hiking, but unfortunately time does not allow all these activities. He can be reached via e-mail at bashansh@il.ibm.com

## David E. Bellagio

**David E. Bellagio** has been involved in the software development community for the past 30 years, ever since he caught the programming bug growing up in Healdsburg, California. David holds a B.S. and an M.S. in computer science, with honors, from California State University at Chico. David has worked at various companies, including Computer Sciences Corporation, Tandem Computers, Automatic Data Processing, and Hewlett Packard. He started with Rational Software as a technical field representative in the Pacific Northwest in early 1998.

David currently is a Worldwide Integration Engineering Architect at IBM Rational Software in charge of developing and deploying integrated solutions to customers around the world. He has worked on-site with numerous customers to define and manage successful deployments of Rational Software solutions. David has presented the following topics at Rational user conferences:

“Building Software with Clearmake on Non-ClearCase Hosts” (Lexington, MA, 1995)

“ClearAdmin: A Set of Scripts, Processes, and Techniques for Administrating Large ClearCase Sites” (Lexington, MA, 1996)

“UCM Stream Strategies and Best Practices” (Dallas, TX, 2004)

“Automating Code Integration Activities with ClearQuest ALM, UCM, and Build Forge” (with Shmuel Bashan, Orlando, FL, 2009)

David coauthored *Software Configuration Management Strategies and IBM Rational ClearCase: A Practical Introduction, Second Edition* (IBM Press, 2005).

David currently resides in the state of Washington with his lovely wife and three children. When time allows, David enjoys playing rock music, shooting pool, and brewing fine ales and mead. He can be reached via e-mail at dbellagio@us.ibm.com.

# Managing Agile Projects

Many software projects are moving to Agile methods. This chapter is not about teaching Agile development; you can read about Agile in many articles and books. Instead, it is about realizing Agile with Rational Team Concert (RTC) and ClearQuest.

While Rational Team Concert was developed with Agile methods in mind, Rational ClearQuest is an older product and was developed with more traditional methods in mind. Luckily ClearQuest is highly customizable, so we can develop schemas with a process that meets the modern development environment.

In this chapter we shall explain in brief the Agile method concept, just to set the right context for the rest of the chapter. The Scrum process is becoming more popular with all types of software development projects. We will use RTC's process enactment of Scrum to illustrate some of the methods within this Agile process. We shall later explain how to build a schema with ClearQuest that will help Agile teams and stakeholders manage their projects smarter to help produce better products.

ClearQuest users using Application Lifecycle Management (ALM) will learn how to configure a project to meet the needs of Agile teams.

## 11.1 Defining Agile Development

There is no single definition of what Agile development is. There are some principles that many agree upon, and teams can adapt them as it suits their organization. Scott Ambler (see "Resources" at the end of this chapter) defines Agile software development as follows:

- Agile is an iterative and incremental (evolutionary) approach to software development
- which is performed in a highly collaborative manner
- by self-organizing teams
- with "just enough" ceremony

- that produces high quality software
- in a cost effective and timely manner
- which meets the changing needs of its stakeholders.

We shall later see how both Rational Team Concert and a ClearQuest schema we propose respond to this definition.

The Agile system development lifecycle is described in Figure 11.1.

## 11.2 Agile and Scrum in a Nutshell

Scrum is a framework for managing Agile software development projects. A Scrum project starts with a definition of the items that the system should include and address, including functionality, features, and technology. These items are often called *requests*. The list of requests is called the *product backlog* (or project backlog). The requests can be of various types: textual requirements, use cases, test cases, stories, defects, enhancement requests, and so on. The requests are likely captured in an external tool such as Rational Requirements Composer (RRC) and are referenced or linked from the change management tool.

Requests can be submitted by any team member, affected users, or stakeholders. The request content is elicited from various sources, but the requests are prioritized by the product owner only and not by the development team. The product backlog is dynamic; it changes and evolves as the project advances.

During the first phase of the project, often called *warm-up*, the requirements are analyzed by SMEs to determined feasibility, cost, effort, and risk. This phase is done with the close participation of the stakeholders. The analysis gives the stakeholders a better basis for setting their priorities.

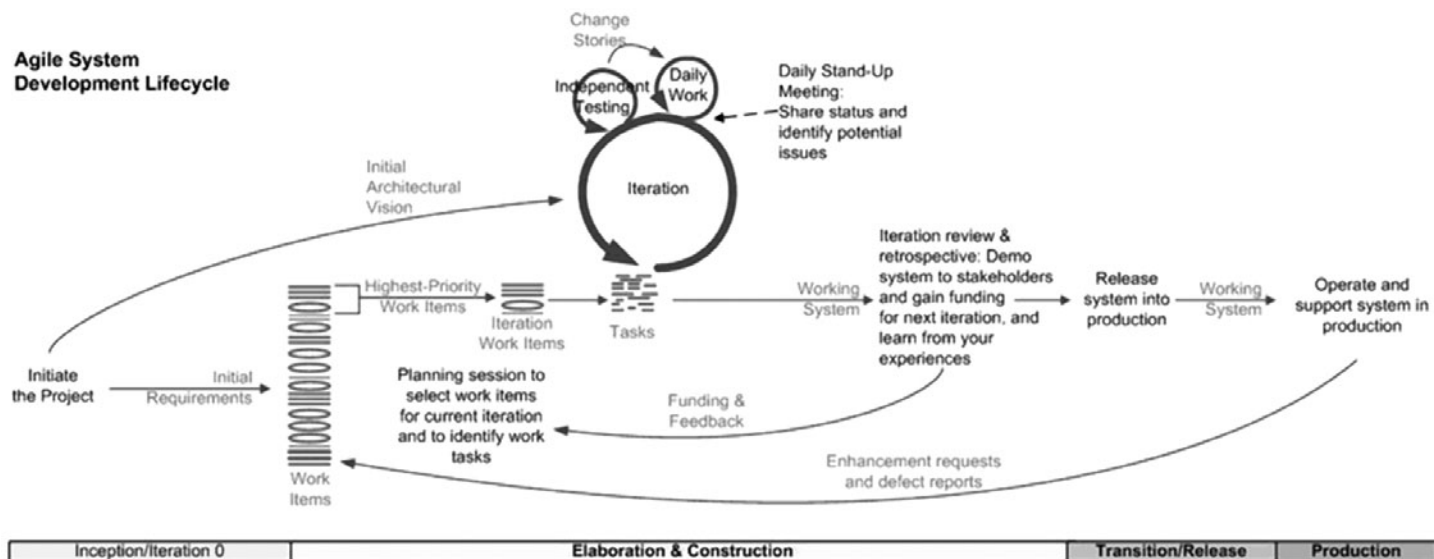
Agile projects are divided into iterations; the iteration has a fixed duration of a few weeks, usually two to four. In Scrum the iteration is called a *sprint*.

The Scrum team's responsibility is to take the requirements from the backlog and develop a usable product or component that has real value to the project, within the sprint period. The team is cross-functional and performs all activities related to the delivery. The team as a whole is responsible for delivering the requirements.

The sprint starts with a team planning meeting. Each team takes one or more top-priority requests from the product backlog, as many as they think they can develop and deliver during the sprint. A sprint must finish with delivery of a new, executable product; thus each sprint consists of all lifecycle disciplines: design, development, testing, and so forth.

The team creates a list of activities (or tasks) from the requirement(s) they have selected. This is called the *sprint backlog*. The activities represent the way the team decides to implement the requirements. Each activity is assigned to a team member (or sometimes to team pairs).

Each Scrum team is autonomous; the team decides how to develop work products from a requirement. Teams are organized in a way that they could be autonomous, thus having expertise in various domains. The team decides which of the Agile development methods to use—XP, pair



**Figure 11.1** The Agile System Development Lifecycle (copyright Scott W. Ambler)

programming, test-driven development, or another method. They decide how much modeling to do. However, the team must adhere to regulations, standards, and rules that the organization has set.

An important role within the Scrum process is the Scrum Master. This role facilitates and guides the teams in adopting the agreed-upon practices and removes impediments. The Scrum Master is not part of the team and does not give orders to the team.

Every day the team gathers for a short (15-minute) stand-up meeting called a Daily Scrum Meeting. The purpose of the meeting is to share status and identify potential issues. During the Daily Scrum Meeting the team progress is reviewed and impediments are identified for removal by management. The Scrum Master facilitates the daily meetings, tracks progress, and works to resolve any inhibitors raised by team members.

Each Scrum member briefly reports on three items:

- What was accomplished since the previous meeting?
- What is planned to be accomplished before the next meeting?
- What prevents the member from accomplishing the activities?

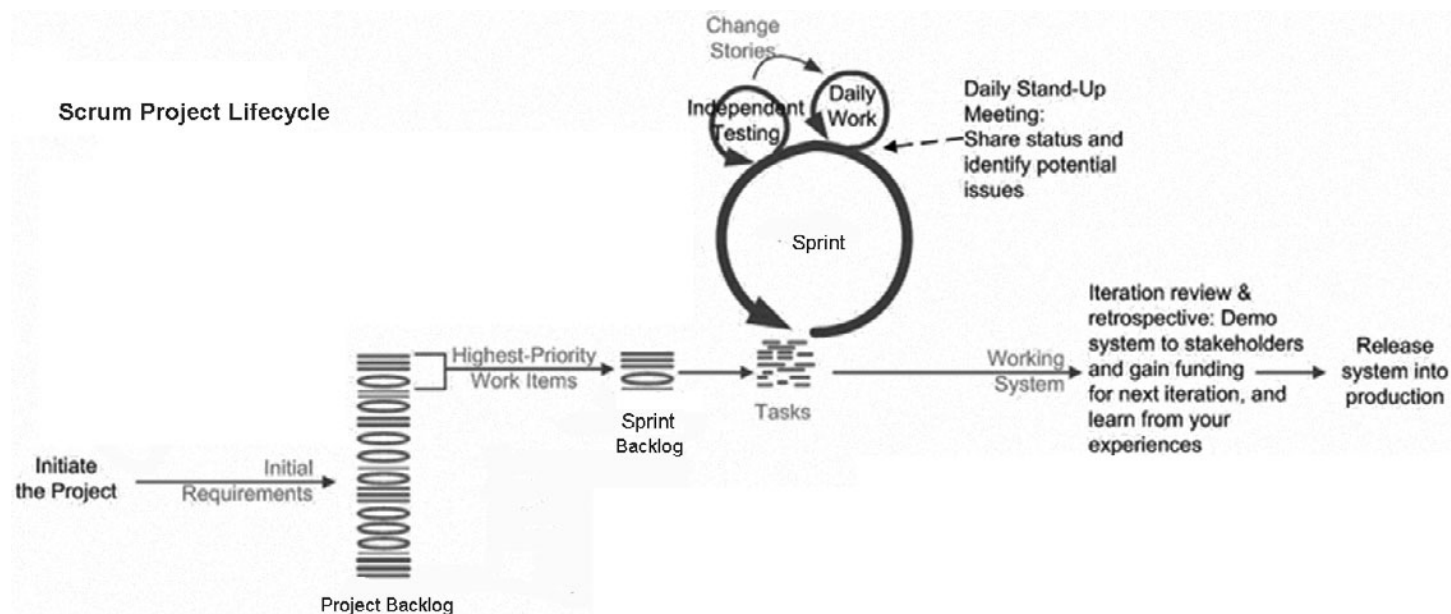
The report should focus on information that helps other team members gain knowledge, learn a lesson, or contribute from their experience. Important practices in the meeting are honesty and transparency.

At the end of the sprint, the team is gathered for a Sprint Review Meeting with the stakeholders. The team reviews the work that was completed and that was not completed. The products developed are demonstrated to the stakeholders to examine their value. They will make a decision about whether to make use of the products and obtain funding for the next iteration. Additional elements of the meeting are to learn from the experience in order to make an improvement for the next iteration. Some teams conduct a different meeting for that purpose; members give their opinions on what went well and what needs improvement. This meeting is called a Sprint Retrospective.

The Sprint Review usually results in some adaptation of the product backlog. Enhancement requests are added, defects are submitted, and maybe new features are introduced. The remaining items in the sprint backlog are moved to the product backlog. The team can add an activity, for example, to learn and experiment with a new technology, or to perform more performance tests. Stakeholders and product owners may decide to change the priorities of some backlog requirements.

Now a new sprint starts again with teams selecting top-priority requests for development. The sprint cycles continue until the stakeholders think they have enough value and quality to release a product. This stage is called the Release Iteration or the End Game. During this iteration final system testing and acceptance testing are performed. Stakeholders may request some defect fixing. The team finalizes system and user documentation. Users and administrators are trained, and the system is deployed to production.

The Scrum project process is often described using the schema illustrated in Figure 11.2.

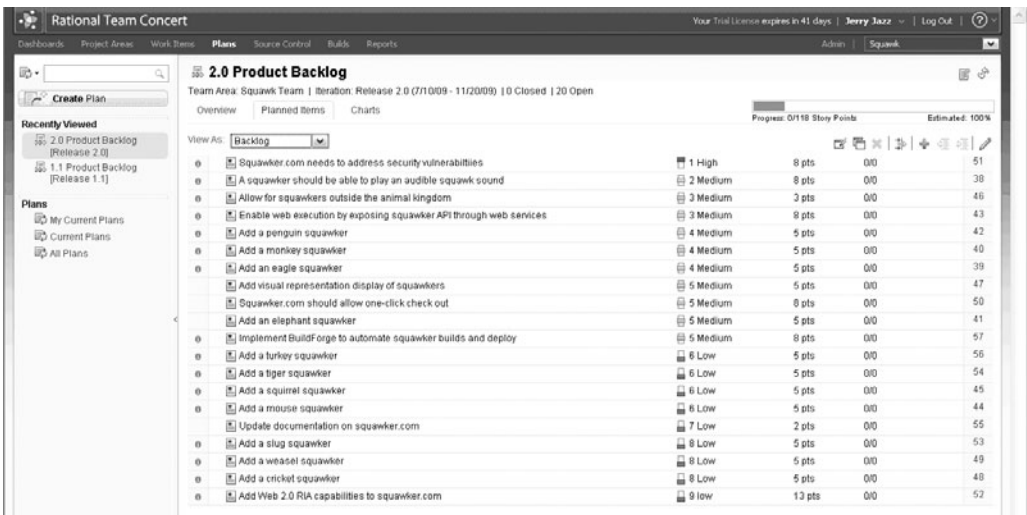


**Figure 11.2** The Scrum Project Lifecycle (copyright Scott W. Ambler)



## 11.3 Realization with Rational Team Concert

Out of the box, Rational Team Concert has a process template that enacts the Scrum process. This chapter will discuss what Scrum is in terms of how Rational Team Concert realizes it. By using a tool's out-of-the-box process, your organization can be more Agile, as you will spend less time designing, creating, and testing a custom solution. Within Rational Team Concert, the Scrum project starts with a definition of all the items that the system should include and address, including functionality, features, and technology. These items are called *stories*. Rational Team Concert has built in an Agile planning feature through its Web interface. The lists of stories are presented in a product backlog plan. This plan interface is the main focus during the team planning meeting to decide what to work on in the first sprint. The product backlog list is shown in Figure 11.3.



**Figure 11.3** The product backlog of Release 2.0

The user stories can be submitted by any team member, affected users, or stakeholders. They are submitted against the backlog category and will show up in the product's backlog plan and are not assigned to any individual yet. The story content is elicited from various sources. A work item of type Story is shown in Figure 11.4.

The product backlog is dynamic; it changes and evolves as the project advances. Certain items may become more important than others during a sprint. Sometimes this process of prioritizing is called *ranking*. With Rational Team Concert you can easily drag stories around to rank them relative to their position in the list. You also get a visual display of unranked items. All stories should be ranked to make sure they are not missed. A product backlog ranked list is shown in Figure 11.5.

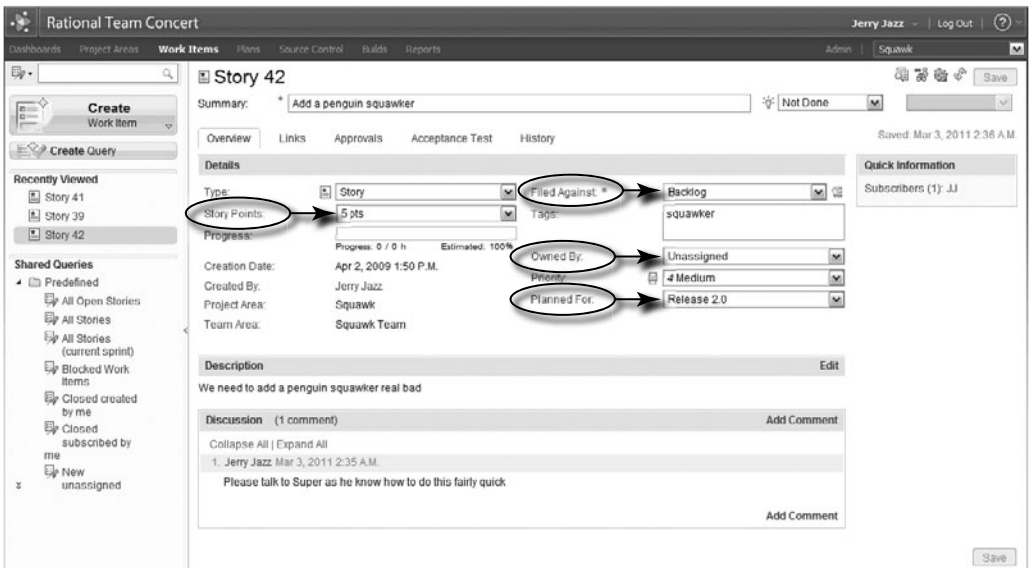


Figure 11.4 The Story work item

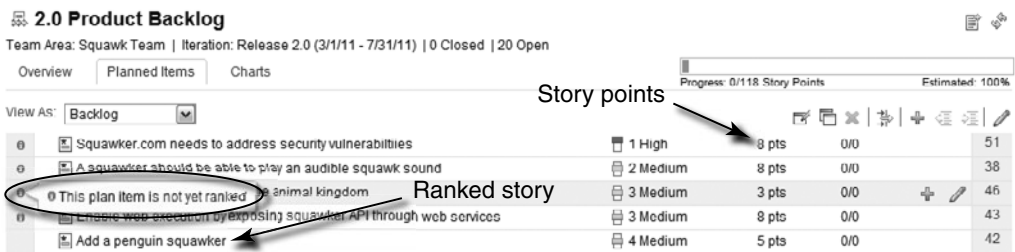


Figure 11.5 Ranking user stories

Within Rational Team Concert you have the ability to attach complexity of effort to a story. The attribute of the Story work item used to convey complexity is Story Points. The larger the Story Points, the more difficult it will be to implement the Story.

In Rational Team Concert you can set up multiple teams working on multiple releases; each release plan has its own current sprint (see Figure 11.6).

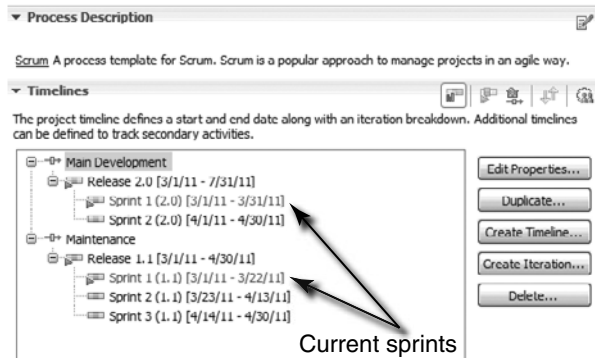


Figure 11.6 Sprints of different releases

Figure 11.7 shows the backlog and the sprints (iterations). The backlog contains a list of work items (in this figure of type Story) that should be assigned to teams in a specific sprint.

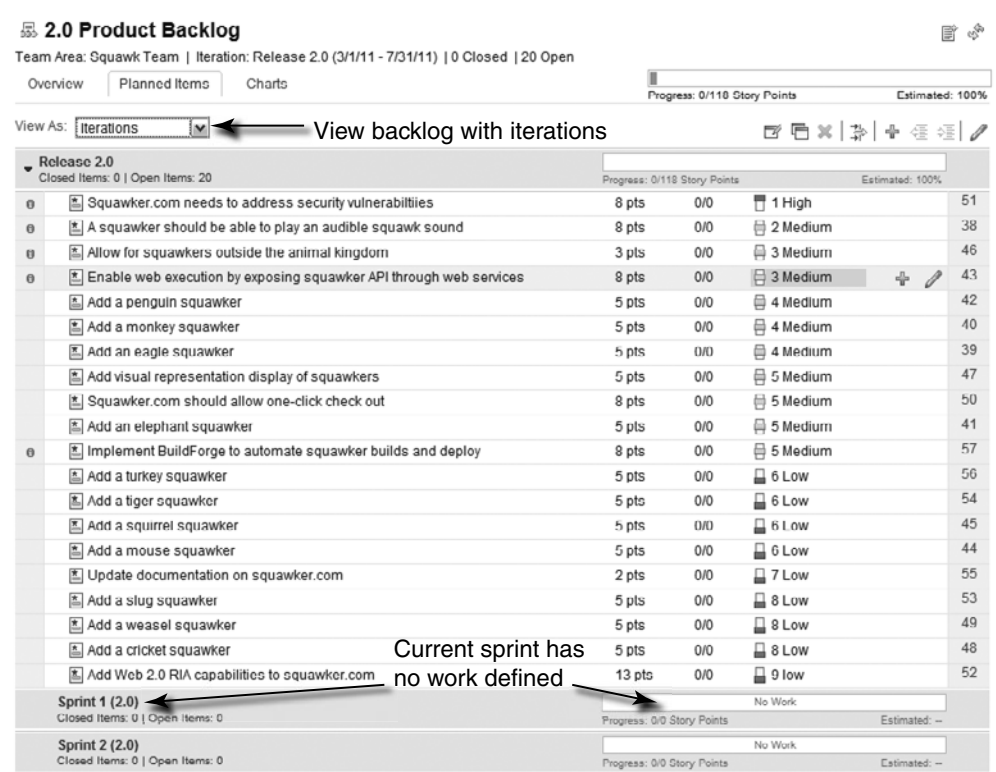


Figure 11.7 Release 2.0 backlog showing iterations

In Rational Team Concert you facilitate sprint planning meetings by simply dragging and dropping a story from the product backlog to your sprint; this is shown in Figure 11.8.

**2.0 Product Backlog**

Team Area: Squawk Team | Iteration: Release 2.0 (3/1/11 - 7/31/11) | 0 Closed | 20 Open

Overview | **Planned Items** | Charts

Progress: 0/118 Story Points | Estimated: 100%

View As: Iterations

Release 2.0		Progress: 0/118 Story Points		Estimated: 100%	
Closed Items: 0   Open Items: 20					
Implement BuildForge to automate squawker builds and deploy	0 pts	0/0	4 Medium		57
Add a penguin squawker	5 pts	0/0	4 Medium		42
Add a monkey squawker	5 pts	0/0	4 Medium		40
Add visual representation display of squawkers	5 pts	0/0	5 Medium		47
Squawker.com should allow one-click check out	8 pts	0/0	5 Medium		50
Add a tiger squawker	5 pts	0/0	6 Low		54
Add a squirrel squawker	5 pts	0/0	6 Low		45
Add a mouse squawker	5 pts	0/0	6 Low		44
Update documentation on squawker.com	2 pts	0/0	7 Low		55
Add a slug squawker	5 pts	0/0	8 Low		53
Add a cricket squawker	5 pts	0/0	8 Low		48
Add Web 2.0 RIA capabilities to squawker.com	13 pts	0/0	9 Low		52
Sprint 1 (2.0)		Progress: 0/47 Story Points		Estimated: 100%	
Closed Items: 0   Open Items: 8					
Enable web execution by exposing squawker API through web services	8 pts	0/0	1 High		43
Squawker.com needs to address security vulnerabilities	8 pts	0/0	4 Medium		51
Add an eagle squawker	5 pts	0/0	4 Medium		39
A squawker should be able to play an audible squawk sound	8 pts	0/0	4 Medium		38
Allow for squawkers outside the animal kingdom	3 pts	0/0	4 Medium		46
Add an elephant squawker	5 pts	0/0	5 Medium		41
Add a weasel squawker	5 pts	0/0	6 Low		49
Add a turkey squawker	5 pts	0/0	6 Low		56
Sprint 2 (2.0)		Progress: 0/0 Story Points		Estimated: -	
Closed Items: 0   Open Items: 0					
No Work					

**Figure 11.8** Assigning stories from the product backlog to a sprint

Rational Team Concert can easily show a plan of any iteration. Once a team creates its Sprint backlog, it can be easily communicated and worked further through the Web planning interface of Rational Team Concert. Figure 11.9 shows sprint backlog stories ordered by priority.

Through the use of Rational Team Concert's Agile planning through the Web interface, you can easily create child tasks from a story. You can drag and drop tasks to make them children of stories, and promote and demote them as needed; this is demonstrated in Figure 11.10.

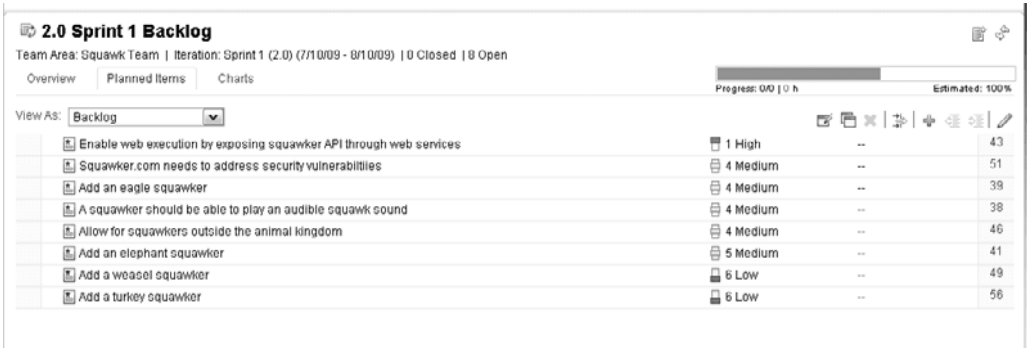


Figure 11.9 The sprint backlog

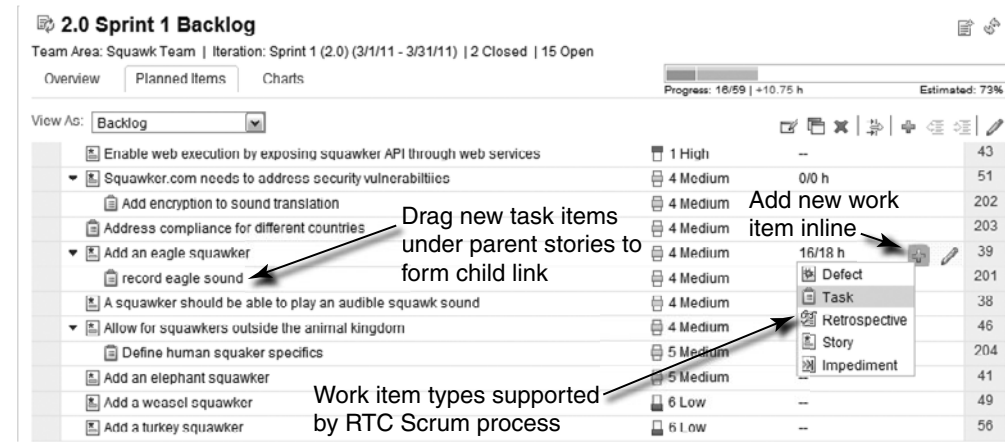


Figure 11.10 Working with the sprint backlog

In order to make sure work gets done, you want to assign tasks to individuals. Rational Team Concert supports Agile assignment of work through drag and drop, making it easier to use the tool to run a sprint planning meeting. As you are assigning work to people, you can also easily enter the amount of task time remaining. This is used to calculate the burndown metric. This work breakdown view of the backlog and the progress bar is shown in Figure 11.11.

Within Rational Team Concert, the Scrum Master role and the other Scrum roles are part of the out-of-the-box process supporting Scrum as shown in Figure 11.12. The administrator can define new roles as required.

**2.0 Sprint 1 Backlog**

Team Area: Squawk Team | Iteration: Sprint 1 (2.0) (3/1/11 - 3/31/11) | 2 Closed | 15 Open

Overview | **Planned Items** | Charts

View As: **Work Breakdown** |

Progress: 10/59 | +10.75 h | Estimated: 73%

**Annotations:**

- View backlog as work breakdown (points to View As dropdown)
- Click to set time remaining (used for burndown) (points to clock icon)
- Drag work item to team member to assign (points to team member icon)

Team Member	Item	Priority	Load	Estimated	Points
Jerry Jaz77	Squawk.com needs to address security vulnerabilities	4 Medium	0/0 h	51	51
	Add design docs	1 hour	6 Low	207	207
	change doc	-	Unassigned	217	217
Student01	Address compliance for different countries	1 week	4 Medium	203	203
	Add an eagle squawker	4 Medium	16/18 h	39	39
Student02	Allow for squawkers outside the animal kingdom	4 Medium	0/0 h	46	46
	Add an eagle squawker	4 Medium	16/18 h	39	39
Student03	Squawk.com needs to address security vulnerabilities	4 Medium	0/0 h	51	51
	Squawk.com needs to address security vulnerabilities	4 Medium	0/0 h	51	51
Student04	Squawk.com needs to address security vulnerabilities	4 Medium	0/0 h	51	51
	Squawk.com needs to address security vulnerabilities	4 Medium	0/0 h	51	51
Student05	Enable web execution by exposing squawker API through web services	1 High	--	43	43
	A squawker should be able to play an audible squawk sound	4 Medium	--	38	38
Unassigned	Add an elephant squawker	5 Medium	--	41	41
	Add a turkey squawker	6 Low	--	56	56
Unassigned	Add a weasel squawker	6 Low	--	49	49
	I can't get system installed	-	Unassigned	205	205

Figure 11.11 Assigning tasks to individuals

**Roles**

Each project area and each team area can define a set of **roles**. The defined roles are visible in the area where they're declared and in all child areas. Roles defined in the project area can be assigned to users for the whole project area or they can be assigned in any team area. Roles defined in a team area can similarly be assigned in that team or in any child team. The ordering of roles in this section determines how they will be ordered in other sections of the editor, but it does not affect the process runtime.

**Defined Roles**

- Product Owner
- Scrum Master
- Team Member
- Stakeholder

**Role Details**

Identifier: ScrumMaster

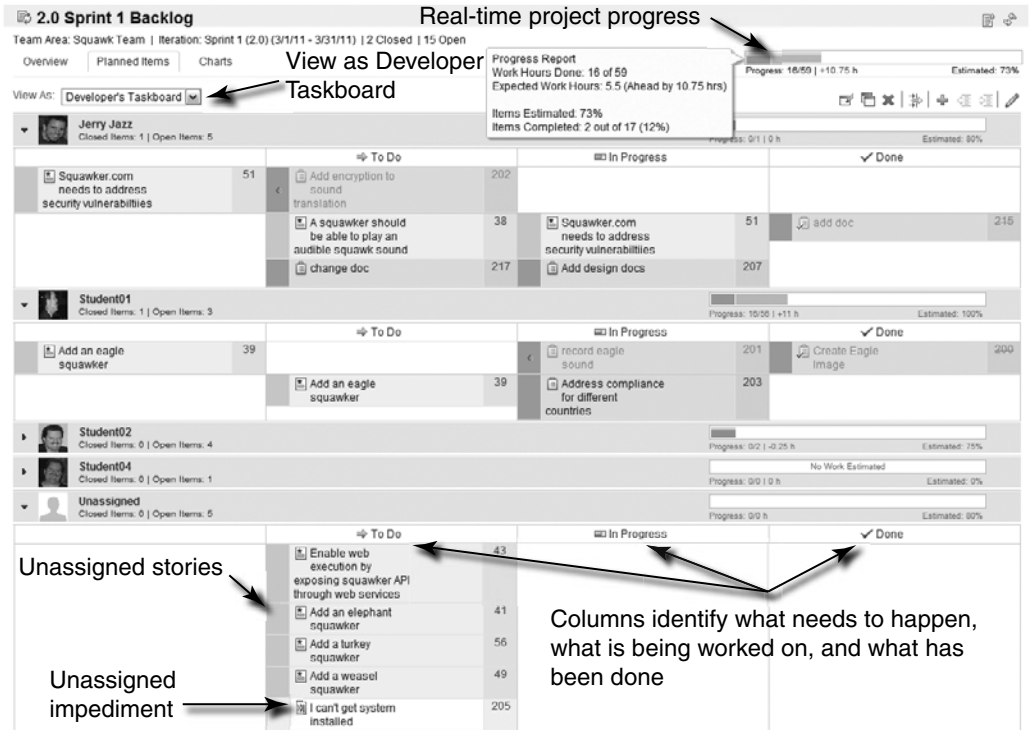
Name: Scrum Master

Cardinality: ☒ single ☐ many

Description: The person responsible for the process.

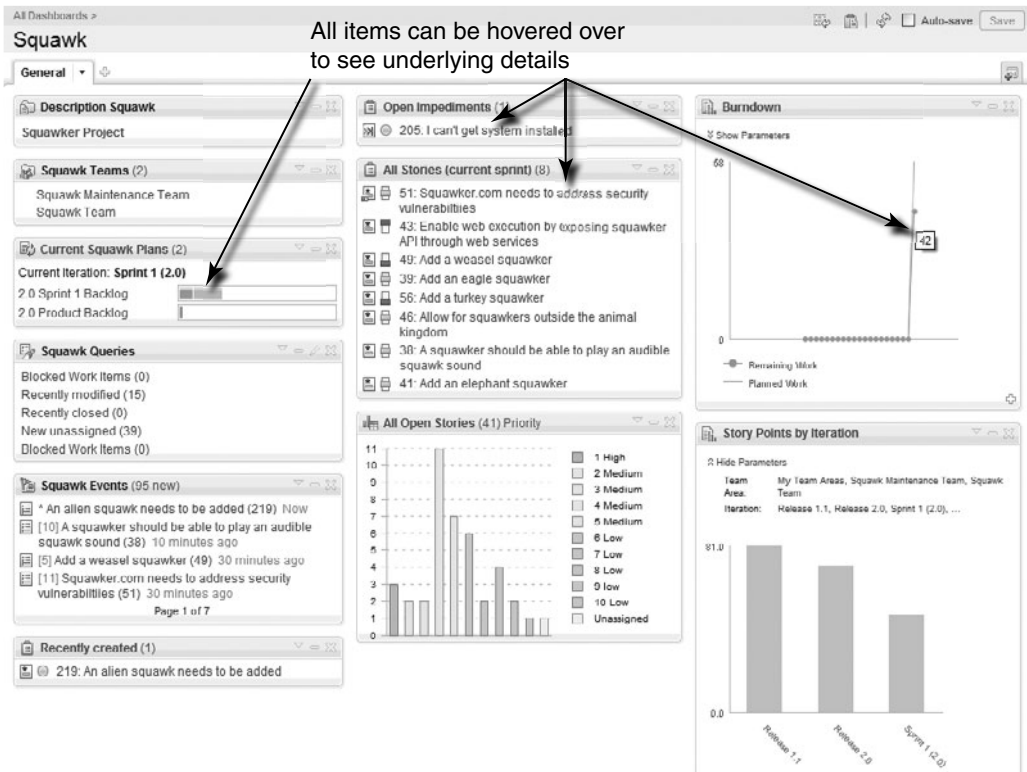
Figure 11.12 Process roles in Scrum

In Rational Team Concert's Scrum process, impediments are implemented as work items and follow a workflow of open → resolved, so they can be submitted, assigned, and managed just like any other work item. Rational Team Concert also provides another useful view for Agile planning through the Web interface called the Developer Taskboard. This view is perfect for the Daily Scrum Meeting as you can easily see what team members are working on, what they have completed, and other data. The Developer Taskboard view is shown in Figure 11.13.



**Figure 11.13** Using the Developer Taskboard in the Daily Scrum Meeting

Another Agile communication vehicle within Rational Team Concert is the dashboard. This is a way that stakeholders and executives can keep informed on the progress of any project. Much of the information contained within Rational Team Concert can be presented easily through feeds to a dashboard. Any Rational Team Concert user (assuming the user has authorization) can set up a dashboard to collect and present items of interest. A sample Rational Team Concert dashboard is shown in Figure 11.14.



**Figure 11.14** Project dashboard showing real-time status of events of interest

One of the types of work items that Rational Team Concert supplies out of the box in the Scrum process is a retrospective. You use this work item to log and track the issues that you discuss during the Sprint Retrospective.

## 11.4 Realization with ClearQuest

In section 11.2, “Agile and Scrum in a Nutshell,” we described the Agile process and the Scrum in particular. Unlike Rational Team Concert, which has a built-in process template to support Scrum, ClearQuest does not have an Agile built-in schema. In this section we shall explain how to build a schema to support Agile projects. Reading the process description in section 11.2, we can identify data objects and workflow scenarios. Now let us take out the data objects from the description. These are

- Request
- Product backlog of requests



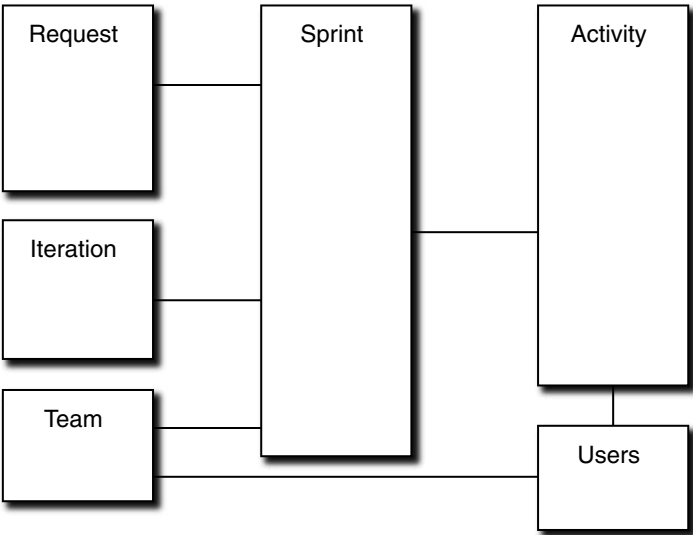
- Sprint
- Activity
- Sprint backlog of activities
- Team
- Iteration

Let's discuss each of these data objects a bit more.

- A request (or CR or any other name that may fit your environment) is realized by a state-based record type. We shall describe the fields of this record type in the next section, but one important field should be the RequestType. Types could be Defect, Enhancement, Feature, Story, Test Case, and so on. Another important field is Priority; the team will select Requests to implement in the iteration based mainly on priority.
- The product backlog of requests and sprint backlog of activities are lists of work items. We do not need to create a special object for these; the reason is that the logs will be realized with queries whose result set is the backlog. The product backlog is a query on the Request record type that filters all of the analyzed requests (ready to be selected for the sprint), sorted by priority. The sprint backlog is a query on the Activity record type that filters all of the opened activities (not completed) that are referenced from a specific sprint, sorted by priority.
- A sprint is realized by a state-based record type. This record type will have fields of type Reference\_List to the Requests and the Activity record types, and fields of type Reference to the Team and Project record types.
- An activity (or task or work item or any other name that may fit your environment) is realized by a state-based record type. As stakeholder requests are usually high-level and nontechnical, the team will break down requests into activities. Each activity will be assigned to a team member.
- The team is realized by a stateless record type. The Team record type contains a list of team members, specific roles in the team such as Team Leader, and users having a role.
- Iteration is realized by a stateless record type. The Iteration record type contains the iteration name, the start date, and the end date.

The record types and their relationships are described in Figure 11.15.

We have realized some of the data objects with state-based record types and some with stateless record types, and we have realized the product backlog and the sprint backlog with queries.



**Figure 11.15** Entities relation diagram for the Scrum schema

**11.4.1 Required Data**

In this section we describe the fields for each record type that are essential for the solution. It is assumed that each implementation will include additional fields based on products developed, organizational culture, regulations, and other considerations.

Table 11.1 describes the suggested fields for the Request record type.

**Table 11.1** Request Suggested Fields

Field Name	Field Type	Comments
Headline	Short_String	
Description	Multiline_String	
CR_Type	Short_String	Closed choice list of request types
Priority	Short_String	
Requestor	Reference	Reference to the user submitting the request
RequestForProject	Reference	Reference to the project record (optional)
Activities	Reference_List	List of the activities this request breaks down to
Iteration	Reference	Reference to the Iteration record
EstimatedEffort	Integer	Estimated effort in hours to deliver the request; mandatory in the analysis

Figure 11.16 is a screen shot of the Request record, with several of the fields listed in Table 11.1.

The screenshot shows a software window titled "View CR scrum00000024". It has two tabs: "Request" (selected) and "Related Items". The "Request" tab contains several input fields:
 

- id**: A text box containing "scrum00000024".
- Headline**: A text box containing "The system shall allow printing of all file types."
- CR\_Type**: A dropdown menu with "Feature" selected.
- Priority**: A dropdown menu with "2-Medium" selected.
- State**: A text box containing "InSprint".
- Description**: A large text area containing "User will be able to printing the current file to a specific physical printer."
- Iteration**: A dropdown menu with "M2" selected.
- Estimated Effort**: A text box containing "44".

 On the right side of the window, there are four buttons: "OK", "Cancel", "Print Record", and "Actions" (with a dropdown arrow).

**Figure 11.16** Request record: main tab

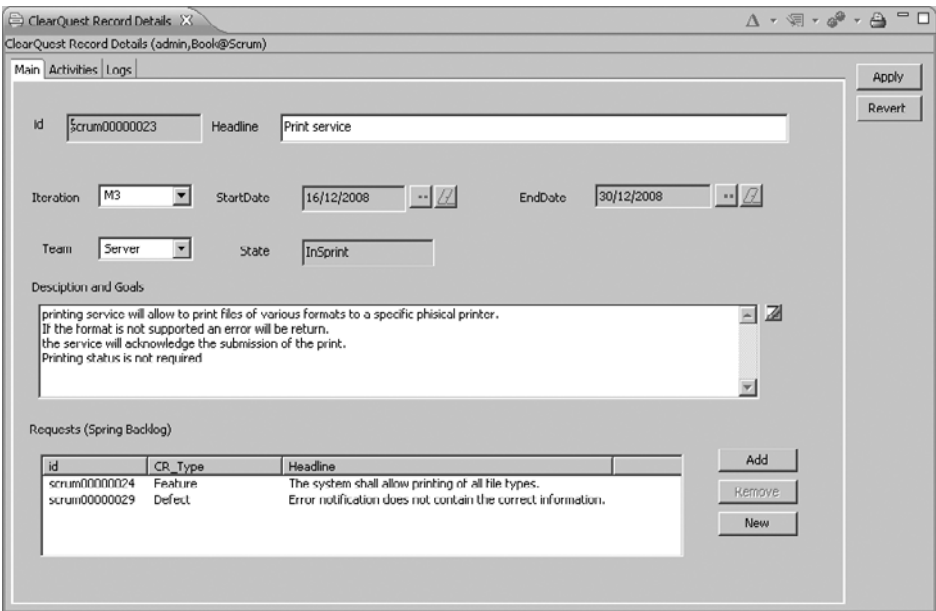
Table 11.2 describes the suggested fields of the Sprint record type.

**Table 11.2** Sprint Suggested Fields

Field Name	Field Type	Comments
Headline	Short_String	
Description	Multiline_String	
Iteration	Reference	Reference to the Iteration record
StartDate		Display only, derived from Iteration.StartDate
EndDate		Display only, derived from Iteration.EndDate
Requests	Reference_List	Reference to the Request record; optionally include back reference
Activities	Reference_List	List of the activities this sprint breaks down to
Team	Reference	Reference to the Team record
Review	Multiline_String	

**Note** We have not included the back reference for the Requests and Activities fields. It is not required for the Scrum process as suggested in this chapter. However, it may ease the creation of several queries and reports.

Figure 11.17 displays the Sprint record main tab; it shows the sprint details, the responsible team, iteration name, end date, and other information. You can see the list of requests that will be realized by the team in this sprint, in this case one new feature to develop and one defect to fix.



**Figure 11.17** Sprint record: main tab

Table 11.3 describes the suggested fields of the Activity record type.

**Table 11.3** Activity Suggested Fields

Field Name	Field Type	Comments
Headline	Short_String	
Description	Multiline_String	
ActivityType	Short_String	Closed choice list of work items/activity types

*continues*

**Table 11.3** Activity Suggested Fields (*Continued*)

Field Name	Field Type	Comments
Priority	Short_String	
Owner	Reference	Reference to the user who is the solution provider
DueDate	Date_Time	Optional
ResolutionDescription	Multiline_String	
ActualDate	Date_Time	Optional
EstimatedEffort	Integer	Estimated effort in hours to deliver the request; mandatory in the analysis
ActualEffort	Integer	Actual effort in hours (optional)
UnitTest	Short_String	Unit test name; may be an automated script name

Figure 11.18 is a screen shot of the Activity record main tab, with several of the fields listed in Table 11.3.

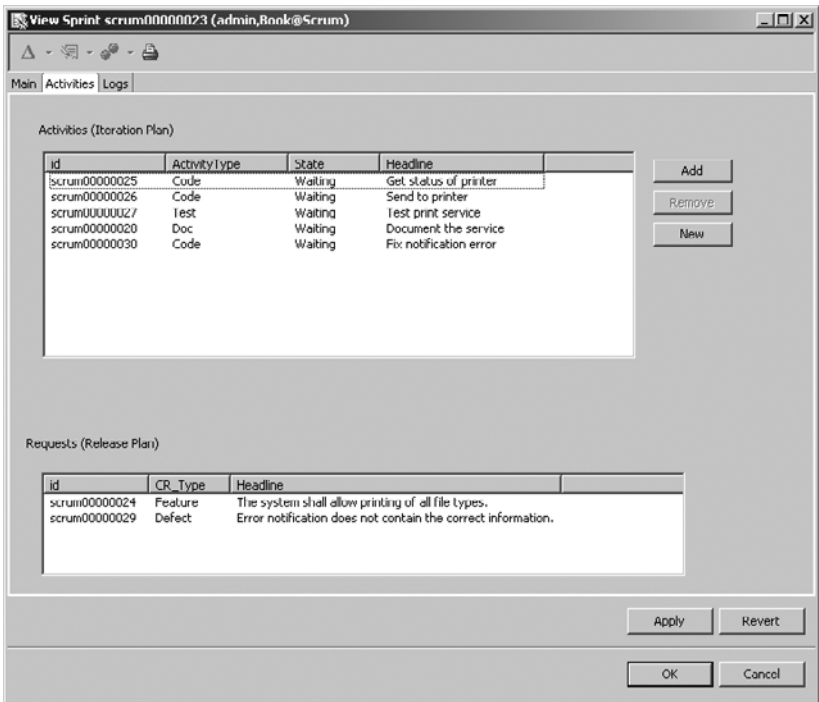
The screenshot shows a 'View Activity' dialog box with the following fields and values:

- id:** scrum00000025
- Headline:** Got status of printer
- ActivityType:** Code
- Priority:** 2 Medium
- State:** Waiting
- Description:** (empty text area)
- DueDate:** 00:00:00 2009\_08 18 11
- EstimatedEffort:** 5
- Owner:** Anath

Buttons on the right: OK, Cancel, Print Record, Actions.

**Figure 11.18** Activity record: main tab

Figure 11.19 displays the Activities tab in a Sprint record. The team has created five activities of different types, to realize the two requests that are shown in the Requests field.



**Figure 11.19** Activities and Requests related to a sprint

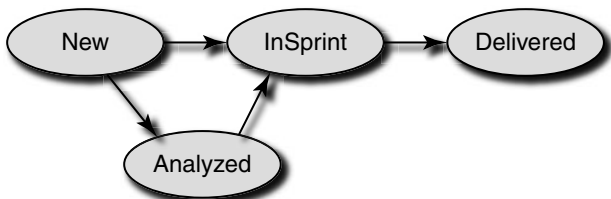
We have seen in this section the record types and the fields that construct the Scrum schema.

### 11.4.2 Understanding the Workflows of Each Record Type

After creating the three state-based records types and the fields in each one, we need to define the state machine for each record type. In the next section we describe the workflow for each record type.

#### 11.4.2.1 Request

The workflow for the Request record type depends a lot on the organization, the stakeholders, and the regulations enforced. We propose the flow shown in Figure 11.20.



**Figure 11.20** The Request record type workflow

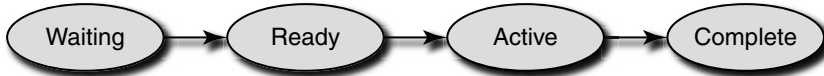
The Request is submitted to the New state and analyzed to identify feasibility, effort, priority, possible impacts, risks, and so on. After it is analyzed and priority is given by the stakeholders, it can be picked by a team to be developed in a sprint. At the end of the sprint during the Retrospective (review meeting) the deliverables are evaluated. If the deliverables are found to meet the stakeholder request and have the desired quality, the Request can be moved to the Delivered state.

In some cases a Request can be moved from the New state directly to the InSprint state, for example, in the case of a defect or a request submitted by the team. In either case the Request must get a priority value by the stakeholder.

#### 11.4.2.2 Activity

The workflow for the Activity record type is similar to the CMBaseActivity record type of the UCM package. We suggest that this record type be integrated with your version control system. If you are using ClearCase, add the UCM package to the schema and enable the Activity record type to the UCM package. The Request and the Sprint record types should not be UCM-enabled because artifact changes are controlled with the Activity record.

The state machine includes four consecutive states: Waiting, Ready, Active, and Complete (see Figure 11.21).



**Figure 11.21** The Activity record type workflow

#### 11.4.2.3 Sprint

The Sprint record type is used for project management as explained in the previous section. During the sprint planning meeting (or before) the record is created and its state is Submitted. When the sprint starts (the sprint iteration start date is reached), the team performs the action StartSprint which moves the record to the state InSprint. When the sprint ends (the sprint iteration end date is reached) and during the sprint Retrospective meeting the team performs the action Close, which moves the record to the state Closed. So the Sprint record type has three consecutive states: Submitted, InSprint, and Closed (see Figure 11.22).



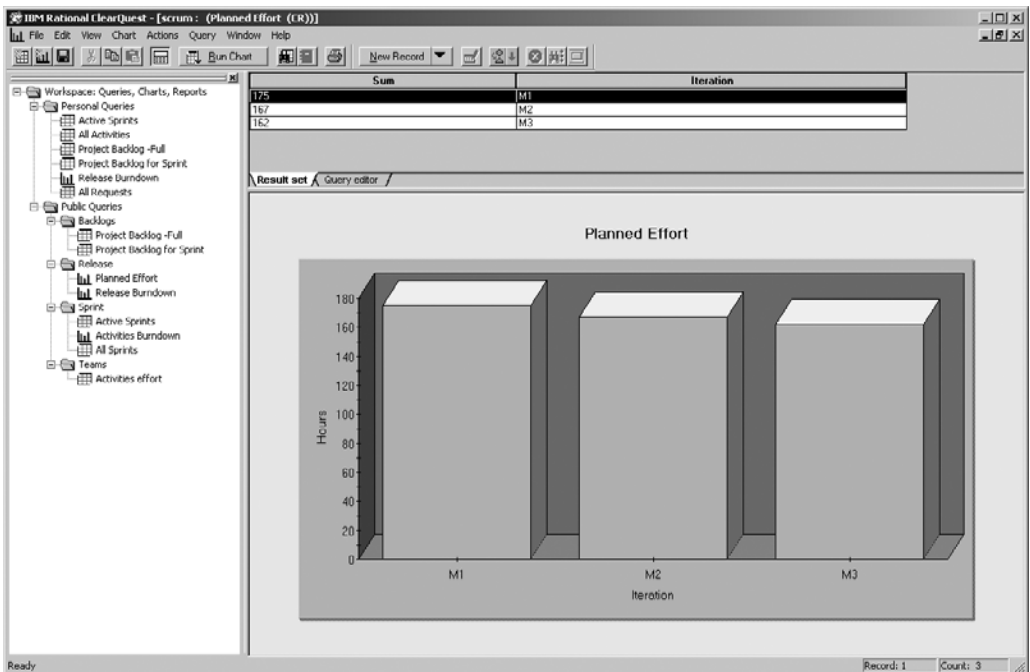
**Figure 11.22** The Sprint record type workflow

In this section we have described the suggested workflow for each of the record types that construct the Scrum schema.

### 11.4.3 Understanding Metrics in Agile Development

We discussed metrics in detail in Chapter 9, “Metrics and Governance”; we include here just a few words on metrics in Agile projects. Metrics measure data of direct business value to the organization. In the repetitive cycles of Agile projects, improvement can bring value, and to improve we need to measure. Working with ClearQuest, we can measure by means of charts and reports provided by the tool.

Figure 11.23 is a screen shot of the ClearQuest client displaying some typical queries and charts for Scrum projects in the workspace. The executed chart shows Planned Effort, and it displays the estimated effort of requests in each iteration.



**Figure 11.23** Planned Effort for Iteration chart

The “classic” release burndown charts and velocity charts cannot be created with the ClearQuest chart wizard, but you can create reports and distribution charts that display the closed activities per iteration, or the actual/estimated effort of activities in an iteration, which will provide similar information. Also, using tools such as Rational Insight will allow you to present burndown and velocity charts.



---

**Creating the “Total Effort per Iteration” Chart** Use the ClearQuest Windows client to create a new chart. For record type select Sprint, and for chart type select Distribution Chart. In the Vertical Axis (Y) select the field Activities.ActualEffort (this is an Integer field); in the Function select Sum. This will sum the effort of all the activities in each iteration. In the Horizontal Axis (X) select the field Iteration.Name.

---

## 11.5 Agile with the ALM Schema

Companies that have decided to adopt the CQ-ALM (ClearQuest Application Lifecycle Management) schema on an enterprise-wide basis may still need to deal with projects and teams that are using an Agile development method. Those companies may ask if they need an additional schema for these teams. Our answer is that they do not; they can use the ALM schema. We shall explain how they can configure an ALM project in an Agile way. We assume that you are familiar to some degree with the CQ-ALM and the ALM terminology. Section 12.4, “ClearCase, ClearQuest ALM, Build Forge Integrated Solution Architecture,” in Chapter 12, “Sample Applications and Solutions,” includes additional discussion and examples of the CQ-ALM schema.

The ALM schema includes three work management state-based record types: Request, Task, and Activity. The Request record is similar to the one described in the previous sections. A Task record is created when the request is approved for development. Priority behavior is mandatory and you should add a field of type Integer for the estimated effort unless you are using CQ-ALM 1.1 where this field is already provided. Approving the request means that it was analyzed by both the technical and the business teams, and it was prioritized by the product owner or the stakeholders. The product backlog is a list of tasks generated by a query that displays all the opened tasks of a given project sorted by priority.

The ALM schema defines project phases and iterations. Agile projects do not use phases, so the ClearQuest admin can create a single phase record and name it with a dummy name such as Iteration or Sprint. The next step is to create iteration records with the numeric values of the iteration. For example, create iteration records and label them with 1, 2, and so forth. Using numeric values is only a suggestion; you can use any method, such as week in the year. When using the system, you will have Iteration 1, Iteration 2, or Sprint 1, Sprint 2, and so forth.

Figure 11.24 shows an ALMTask record assigned to Sprint 2.

Another differentiator between our Agile schema and the ALM schema is the use of roles. Agile projects usually adopt the whole team practice; the skills of the whole team are what matters and team members’ roles are less relevant. So we suggest creating a Role Label record for each team, using names like Team-A, Team-B, and so forth. In the Members tab add the team members to the Members field and the team leader to the Primary field. If relevant to your project add additional role labels such as TeamLeader or ScrumMaster (see Figure 11.25).

In the Scrum schema described in the previous sections we used a record type called Sprint. Do we need to create such a record type in the ALM schema? The answer is no.

The screenshot shows a window titled "Open ALMTask: CQALM00000163". It has several tabs: Task, Project, Related Records, Resolution, History, Comments, and Attachments. The "Task" tab is active. Inside, there's a section "Task's Fix-in Context" with a sub-section "Project Assigned to:" containing a table with columns ID, Name, Category, Release, and Obsolete. The table has one row with values CQALM00000161, Portal, Software, 1.2, and No. Below this are dropdowns for "Phase Assigned to:" (set to Sprint) and "Iteration Assigned to:" (set to 2). At the bottom is a "Notify List:" section with a table with columns login name, email, and full name. On the right side of the window are buttons for OK, Cancel, and a Values dropdown.

ID	Name	Category	Release	Obsolete
CQALM00000161	Portal	Software	1.2	No

Phase Assigned to: Sprint

Iteration Assigned to: 2

login name	email	full name
------------	-------	-----------

**Figure 11.24** ALMTask record assigned to sprint (iteration)

The screenshot shows a window with tabs: Role, Members, Approved Actions, and History. The "Role" tab is active. It contains a section "Project:" with a table with columns id, Name, Category, and Release. The table has one row with values CQALM00000161, Portal, Software, and 1.2. Below this is a "Role Label:" dropdown set to "Blue Team" and a "New Role Label" button. On the right side are buttons for Apply, Revert, Print Record, and an Actions dropdown.

id	Name	Category	Release
CQALM00000161	Portal	Software	1.2

Role Label: Blue Team

**Figure 11.25** ALMRole record assigned to an Agile team

A sprint is a team effort for a given iteration. What we need to do is to link the Request, Task, and Activity to the team and the iteration (see Figure 11.26). Among the many possible solutions we shall mention three:

- Add a field called Sprint to each of the Request, Task, and Activity record types. The team will fill in a value that uniquely identifies the sprint. The field value should be automatically copied when a child record is created. Although this solution is simple, it requires a schema change.
- The second solution uses existing fields. The Task record already has the field Iteration that references the ALMIteration record that has fields such as start\_date, end\_date, status\_label, description, and others. The Activity and the Request record types have references to Task, so for each record we can find the iteration that the record was assigned to. We also need to relate the record to the team working on it; this is done using the ALMRole record as previously explained.
- The third solution is similar to the second one. We previously explained that we gave the phase a dummy value, so instead of using a dummy value we can set the phase name to be the team name. Now we have for the iteration a meaningful value that is the team name and the iteration name. The user will see in the Iteration field Team-A 1, Team-A 2, and so on, which identify the sprint numbers for Team-A.

The screenshot shows the ALMTask record interface. At the top, there are tabs: Task, Project, Related Records, Resolution, History, Comments, and Attachments. The 'Task' tab is selected.

**Headline:** Develop support for Firefox 3.5

**Project Assigned To:** Add Remove

ID	Name	Category	Release	Obsole
CQALM00000161	Portal	Software	1.2	No

**Roles:**

Role Label	Primary
Blue Team	admin

**Description:**

It is required to develop support for additional browsers. The required version are Firefox 3.5, 3.5.1, 3.5.2  
 This is required for all users access.  
 Admin module support is desirable but not mandatory.  
 See the attached specification doc.

**Task ID:** CQALM00000163

**State:** Activated

**Type:** Enhancement

**Priority:** High

**Owner:** admin

**Activities:** Create Activity

Activity ID	Type	Headline	Owner	State	Resolution Code
CQALM00000164	Develop Software	Develop support for Firefox 3.5		Submitted	
CQALM00000165	Test Software	Develop support for Firefox 3.5		Submitted	

On the right side, there are buttons: Apply, Revert, Print Record, and Actions.

Figure 11.26 ALMTask record assigned to a team

The team (field Roles) in Figure 11.26 is automatically selected when the task type is selected. This is defined by creating an ALMWorkConfiguration record with the following fields: Project, Record Type, Type Label, Roles.

The three solutions allow you to create queries to see the sprint status, cumulative effort, status of each activity, defects reported against a specific sprint, and other sprint queries, charts, and reports as required.

During the sprint planning meeting the team creates one or more activities for each task. Each activity is assigned to a team member. The number of child activities that will be created per task and the types of those activities can be defined and set by each team in the ALMWork-Configuration record. This is a powerful and useful feature of the ALM schema.

In the Sprint Review Meeting, if the team found the tested deliverables to have the required quality, the team can Complete (an action) the Activated (a state) Tasks. Now the stakeholder can Accept (an action) the related Requests to release the deliverables and to Close the sprint (see Figure 11.27).

The screenshot shows the 'View ALMRequest: BHALM00000456' window. The 'Request' tab is active. The 'Requester's Found-in Category Context' section shows a table with one row: ID BHALM00000449, Name Work Management, Category CoE, Release AP Release, and Onstate No. The 'Headline' is 'Install color A1 printer' and the 'Description' is 'Installation of color A1 printer (model HAC 236) in building CoHi room 312,'. The 'Request ID' is BHALM00000456, 'State' is Opened, 'Type' is Work, 'Severity' is 4 - Low, 'Owner Name' is Shmuel Beshan, and 'Owner' is heshanish. The 'Tasks' table has two rows: Task ID BHALM00000457, Name Work ..., Project Category CoE, Project Release AP-Release 5 Defects, Type Solution Delivery, Status Completed, and Resolution Code Completed. The 'Submitter' is admin, 'Submitter Domain' is Submitter Domain, 'Submit Date' is 16/12/2010 20:14, and 'Due Date' is 29/12/2010 00:00. An arrow points to the 'Actions' dropdown menu, which is open, showing options: Accept, Modify, Delete, Reject\_Solution, Withdraw, CreateTask, QuestionOrComment, MarkAsDuplicate, Reject\_Request, WorksAsDesigned, Unresolvable, and DuplicateComplete. The 'Accept' option is highlighted.

**Figure 11.27** ALMRequest solution is accepted by the requestor

The ALMRequest record is accepted by the requestor (stakeholder) after the ALMTask is completed with resolution code Completed.

## 11.6 Resources

### 11.6.1 Materials by Scott Ambler

Ambler, Scott, “Agile Modeling—Effective Practices for Modeling and Documentation,” [www.agilemodeling.com](http://www.agilemodeling.com), 2007 (accessed February 23, 2011).

Ambler, Scott, “Scott Ambler’s Articles and Other Writings,” [www.ambysoft.com/onlineWritings.html](http://www.ambysoft.com/onlineWritings.html), 1997–2011 (accessed February 23, 2011).

This page provides links to books and Web-based writings.

Ambler, Scott, “The Agile System Development Life Cycle (SDLC),” [www.ambysoft.com/essays/agileLifecycle.html](http://www.ambysoft.com/essays/agileLifecycle.html), 2005–2010 (accessed February 23, 2011).

### 11.6.2 DeveloperWorks Articles

Pampino, Carolyn, and Robert Pierce, “Application Lifecycle Management with Rational ClearQuest 7.1.0.0,” *IBM developerWorks*, [www.ibm.com/developerworks/rational/library/edge/08/mar08/pampino-pierce/](http://www.ibm.com/developerworks/rational/library/edge/08/mar08/pampino-pierce/), 2008 (accessed February 23, 2011).

Ellingsworth, Millard, and Thomas Starz, “Scrum Project Management with IBM Rational Team Concert Version 2,” *IBM developerWorks*, [www.ibm.com/developerworks/rational/library/09/scrumprojectmanagementteamconcert-1/index.html](http://www.ibm.com/developerworks/rational/library/09/scrumprojectmanagementteamconcert-1/index.html), 2009 (accessed February 23, 2011).

Lee, Kevin A., “Agile SCM and the IBM Rational Toolset,” *IBM developerWorks*, [www.ibm.com/developerworks/rational/library/jun06/lee/index.html?S\\_TACT=105AGX15&S\\_CMP=EDU](http://www.ibm.com/developerworks/rational/library/jun06/lee/index.html?S_TACT=105AGX15&S_CMP=EDU), 2006 (accessed February 23, 2011).

### 11.6.3 Other Information

“Agile Alliance,” [www.agilealliance.org/](http://www.agilealliance.org/), 2011 (accessed February 23, 2011).

“OpenUP,” <http://epf.eclipse.org/wikis/openup>, 2011 (accessed February 23, 2011).

“Scrum Alliance,” [www.scrumalliance.org/](http://www.scrumalliance.org/), 2011 (accessed February 23, 2011).

Wells, Don, “Extreme Programming: A Gentle Introduction,” [www.extremeprogramming.org](http://www.extremeprogramming.org), 2009 (accessed February 23, 2011).

Wikipedia, “Lean Software Development,” [http://en.wikipedia.org/wiki/Lean\\_software\\_development](http://en.wikipedia.org/wiki/Lean_software_development), 2009 (accessed February 23, 2011).

Schwaber, Ken, and Mike Beedle, *Agile Software Development with Scrum* (Upper Saddle River, NJ: Prentice Hall, 2002).

## 11.7 Summary

In this chapter we started with a short explanation of Agile principles and continued with a somewhat more detailed description of the Scrum method. We later explained how Scrum is realized in Rational Team Concert using the provided Scrum process template.

We continued with an explanation of how to create a ClearQuest schema to manage Agile projects and specifically Scrum projects. It is important to mention that the proposed solution can be modified and adapted to each company or project. The same principles can be applied to extend existing schemas. Some examples of metrics derived with ClearQuest charts are explained.

The ClearQuest schema described will be available to download from the IBM developer-Works site. The schema serves as a skeleton and does not pretend to be a complete solution. Use it as a basis for your schema; add fields and hooks to create rules and to automate operations.

In the last section we explained how organizations that use the built-in CQ-ALM schema can configure an ALM project to support Agile teams. We proposed several solutions that require only minimal modifications or no modifications to the schema.

*This page intentionally left blank*

---

# Index

## A

A (active) state type, UCM, 54

Access control

- compared with permissions, 3
- governance and, 288–289
- hook providing security in ClearQuest, 128
- roles in, 127, 144–145, 379
- rules governing, 11
- workflow rules, 46

Accessing records, 114–115

Action hooks, ClearQuest, compared with Jazz Operation Access Control. *See also* Hooks, Access Control

- behavior or extensions, 3
- Commit hook. *See* Hooks, Commit
- Initialization hook. *See* Hooks, Initialization
- Notification hook. *See* Hooks, Notification
- Validation hook. *See* Hooks, Validation

Actions

- comparing ClearQuest action with Jazz action, 2
- customizing work items, 15
- dedicated, 70–76
- in Jazz workflow, 67
- hooks. *See* Action hooks
- reassignment, 70
- roles in automation of, 127
- rules governing, 11
- workflow rules, 46

ActiveX

- applets, 12
- integration with ClearQuest and, 171–172
- supported in BASIC not Perl, 261

Activities

- describing ClearQuest data objects for Agile process, 338
- subclasses of, 5–6
- workflows in activity diagrams, 44

Activity record type

- in ALM schema, 346, 348
- suggested fields for, 341–343
- workflows of, 344

ACWP (actual cost of work performed), 280

Administrative activities, 5

Administrators

- dependent integration of IBM administrator with ClearQuest, 158
- implementing roles with ClearQuest groups, 133
- performance benchmarking by Web administrators, 200
- training, 206

Agile development

- in ALM schema, 346–349
- applying Agile practices to requirements, 28
- in ClearQuest. *See* ClearQuest, Agile schema in defining, 325–326
- in Jazz. *See* RTC (Rational Team Concert), Scrum process in
- overview of, 325
- resources for, 350
- Scrum framework for, 326–329
- summary, 351
- use of graphs in metrics strategy, 281

Agreement, getting regarding requirements, 28

ALM (Application Lifecycle Management)

- schema. *See also* C/ALM (Collaborative ALM)
- Agile development in, 346–349
- common schema in development, 222
- creating task to fix defect, 370–371
- governance in, 297–298
- in integrated solution architecture, 367–373
- periodic releases in, 371–372
- process control in, 297
- resources for, 195, 383
- roles in, 140–141, 298
- security context in, 297–298
- workflows in, 65–66



ALMActivity, 5, 65, 140, 370  
 ALMRequest, 348–349  
 ALMRole  
   comparing Agile and ALM schema, 347  
   overview of, 140–141  
   stateless record types in ALM security, 298  
 ALMRoleLabel  
   overview of, 140  
   stateless record types in ALM security, 298  
 ALMTask, 347–348  
 Amount of effort. *See* Effort required  
 AMStateTypes, 53  
 Analysis  
   data providing information for, 81  
   in workflow item lifecycle, 50  
 Analysis & Design discipline  
   choosing database, 32–33  
   defining client types, 29–30  
   defining data fields, 35  
   defining infrastructure architecture, 30–32  
   defining workflow, 35  
   design patterns in, 36–40  
   designing user interface (forms), 36  
   overview of, 29  
   resources, 40  
   reviewing/signing off on design models, 40  
   schema high-level design in ClearQuest, 33–34  
   summary, 40–41  
 Applets, performing background operations, 12  
 Applications/solutions  
   ClearQuest example. *See* ClearQuest, example solution  
   external impacts and, 89  
   installing during deployment, 202  
   integrated. *See* Integrated solution architecture (ClearCase, ALM, and Build Forge)  
   Jazz-based C/ALM, 354–356  
   resources for, 382–383  
   summary, 384–385  
 Approval tracking, Jazz, 2  
 AppScan, integration of RQM and RTLM with, 185–187  
 Architecture  
   defining infrastructure architecture, 30–32  
   for integrated solution. *See* Integrated solution architecture (ClearCase, ALM, and Build Forge)  
   JIA (Jazz Integration Architecture), 192–193, 355

Areas  
   defined, 130  
   role implementation and, 133–136  
   user roles in different areas, 131  
 Asset reuse, 191  
 Assign action, notification and, 58  
 Assignees, assigning solution providers, 91  
 Assignment  
   auto-assignment, 145–149, 273–274  
   issues/typical problems addressed by roles, 128–129  
   members to test tasks, 311–312  
   reassignment, 70  
   roles in, 127  
 Association, by field label in Eclipse Designer, 109  
 Attachments  
   data, 100–103  
   HasAttachment hook, 120–121  
   Limit Attachment size hook, 121–122  
   to work items, 11  
 Auto-assignment, of roles, 145–149, 273–274  
 Auto-change state, 58–59  
 Automation  
   in ClearQuest governance, 290  
   in governance, 288  
   in Jazz-based C/ALM solution, 354  
   roles in, 127  
   setting choices based on Multiple role, 146–147  
   setting Responsible based on role object, 147–149  
   setting Responsible based on Single role, 145–146  
   Single roles and, 130

## B

Back Reference field, in object relations, 110–113  
 Balking pattern, 36  
 Baselines  
   fixing defects and, 370–371  
   periodic releases and, 371–372  
 BASIC  
   choosing scripting language and, 261  
   creating new ClearQuest integrations, 167–172  
   storing hooks externally, 249–250  
   use of session variables, 240

BCWP (budgeted cost of work performed), 280  
Benchmarking, performance, 200  
Bill of materials (BOM), in preparing for deployment, 201  
BIRT (Business Intelligence and Reporting Tools)  
    as metrics tool, 286  
    resources for, 303–304  
Blocking assignment, 129  
BOM (bill of materials), in preparing for deployment, 201  
Budgeted cost of work performed (BCWP), 280  
Build engines, RTC integrations, 190–191  
Build Forge  
    continuous build and validation process, 369–370  
    dependent integration with ClearQuest, 161–162  
    in integrated solution architecture, 367–373  
    RQM integrations, 187  
Build reports, in metrics strategy, 284  
Build work items, 6  
Builds  
    continuous build and validation process, 369–370  
    subflow for build approval, 76–78  
Burndown charts, releases and, 345  
Business Intelligence and Reporting Tools (BIRT)  
    as metrics tool, 286  
    resources for, 303–304  
Business rules, 378–379  
Buttons, enabling button hooks in Web development, 274–275

## C

C/ALM (Collaborative ALM)  
    Jazz-based, 354–356  
    overview of, 353–354  
    reports, 304–305  
    resources for, 383  
C (complete) state type, UCM, 54  
Caching choice lists, 267–268  
Capability Maturity Model Integration (CMMI), 83, 103  
Cardinality, roles in Jazz, 130  
Categories  
    customizing test work items, 318

    organizing test information using, 320–321  
    setting permissions for test customization, 321–322  
CCRC (ClearCase Remote Client), 172  
Change management  
    defining change process, 207–208  
    process diagram, 47–48  
    roles in, 131  
    RUP (Rational Unified Process) applied to, 21–22  
    system elements, 28–29  
    Unified Change Management. *See* UCM (Unified Change Management)  
Change requests  
    activities for breaking down into smaller elements, 5  
    documenting in change management database, 28–29  
    roles in, 131  
    types of changes and, 4  
Change\_State actions, ClearQuest  
    overview of, 56  
    subflow for gathering more information, 69  
Chart wizard, 285  
Charts  
    comparing ClearQuest and Jazz customization areas, 17  
    in metrics strategy, 281  
Child Control pattern  
    design, 38  
    implementing, 38, 229–231  
Child records, creating from parent record, 123–126  
Choice list, Clear Quest  
    allowed actions list, 150  
    caching, 267–268  
    compared with Jazz Enumeration, 3  
    creating multiple lists, 268  
    creating tree-like lists, 268–270  
    defining requiredness, 361  
    hard-coded data in, 272–273  
    hooks, 114, 144, 150, 236–238, 270–271  
    improving performance of long lists, 265  
    populating based on role objects, 147–148  
    populating based on roles, 146–147  
    recalculating/invalidating, 265–267  
    in solution example, 358–360  
    for user-defined roles, 150

## Classification of work items

- activities, 5–6
- change requests, 4
- overview of, 4
- project-related work items, 6–7
- test elements, 6

## ClearCase

- ActiveX controls in establishing integration with, 171–172
- CM API and, 172–175
- in integrated solution architecture, 367–373
- integrating with ClearQuest, 156–157
- in managing release promotion, 375
- UCM work projects and, 368–369
- working on test artifacts, 372–373

## ClearCase Bridge

- ClearCase Connector compared with, 188–190
- connecting to RTC, 188

## ClearCase Connector

- compared with ClearCase Bridge, 188–190
- RTC integrations, 188

## ClearCase Remote Client (CCRC), 172

## ClearQuest

- ALM schema. *See* ALM (Application Lifecycle Management) schema
- comparing ClearQuest and Jazz terminology, 2–3
- comparing customization elements and terms with Jazz, 15–17
- customizing record types, 18–19
- databases supported by, 32–33
- design patterns built into, 36
- implementation tasks, 198
- implementing roles with ClearQuest groups, 132–133
- integrations. *See* Integration, ClearQuest
- representation of data in, 117–118
- schema high-level design in, 33–34
- scripts for data representation in ClearQuest, 120
- setting up customizations during deployment, 203–204
- setting up environment during deployment, 202–203
- ClearQuest, Agile schema in
  - metrics, 345
  - overview of, 337–339

requesting required data for, 339–340

suggested fields for Activity record type, 341–343

suggested fields for Sprint record type, 340–341

workflows in, 343–345

## ClearQuest Bridge, 180–183

## ClearQuest Connector

- integration at data-level, 154
- Jazz integration with ClearQuest, 177–180
- RTC integrations, 188–190

## ClearQuest Designer

- comparing Eclipse and Windows versions of, 18–19
- moving user databases, 214–216
- user-defined fields in, 357–358
- viewing database properties, 213–214
- Windows version, 213

## ClearQuest, example solution

- defining choice lists, 358–360
- defining requiredness, 360–363
- SLAs (Service Level Agreements) in, 363–367
- user-defined fields, 356–358

## ClearQuest, governance in

- electronic signatures in, 295–296
- monitoring in, 297
- permissions in, 290–293
- process control and automation in, 290
- security context in, 294–295

## ClearQuest Maintenance tool

- creating test environment with, 214
- QATest, 212

## ClearQuest MultiSite (CQMS), 253–254

## ClearQuest to Project Tracker integration, 158–159

## ClearQuest Test Management (CQTM)

- overview of, 6
- RQM replacing, 307
- state-based record types in tests, 309

## ClearQuest Tool Mentor

- creating test environment with ClearQuest tools, 214–216
- creating test environment with database vendor tools, 212–214
- importing records with references, 208–210
- importing updates, 210–211
- overview of, 208

## ClearVision Subversion, ClearQuest integration with, 176

## Clients

- CCRC (ClearCase Remote Client), 172
  - defining client types, 29–30
  - Eclipse. *See* Eclipse client
  - interactions with resources on server, 173
  - representation of data in ClearQuest and, 117–118
  - Web clients, 119, 252, 276
- Cloning hook, for creating parent from child record, 123–126
- Closing pattern
  - implementing in ClearQuest, 223–224
  - implementing in Jazz, 224–225
  - overview of, 37
  - suggested fields, 222–223
- Closure stage, in workflow item lifecycle, 51
- CM (Configuration Management) API, creating new ClearQuest integrations, 172–175
- CMMI (Capability Maturity Model Integration), 83, 103
- Code generation, list of RTC integrations by category, 191
- Code reuse, 105
- Collaboration
  - in Jazz-based C/ALM solution, 354
  - list of RTC integrations by category, 191
- Collaborative ALM. *See* C/ALM (Collaborative ALM)
- Comma-separated value (CSV) format, importing and, 204
- Communication diagrams, workflows in, 44
- Communication Manager (CM) API, creating new ClearQuest integrations, 172–175
- Components
  - internal impacts on, 88
  - in promotion process, 376–379
  - relationships in integrated solution architecture, 368
- Convert Full\_Name to Login\_Name hook, 122–123
- Corrective actions
  - assigning solution providers, 91
  - data related to, 82
  - documenting, 92
  - prioritization of, 91
- CQ-ALM schema. *See* ALM (Application Lifecycle Management) schema

- CQ CM API JNI (Desktop) provider, 172–173
- CQ CM API WAN (Network) provider, 172
- CQMS (ClearQuest MultiSite), 253–254
- CQTM (ClearQuest Test Management)
  - overview of, 6
  - RQM replacing, 307
  - state-based record types in tests, 309
- Crystal reports, as metrics tool, 285–286
- CSV (comma-separated value) format, importing and, 204
- Custom Section wizard, 319
- Customer role, in defect and change management, 131
- Customizing test work items, in RQM
  - default work items, 317–318
  - execution states, 322–323
  - organizing test information using categories, 320–321
  - overview of, 316–317
  - setting permissions for customization, 321–322
  - testing specific work items, 318–320
- Customizing work items
  - ClearQuest record types, 18–19
  - comparing ClearQuest and Jazz customization elements and terms, 15–17
  - elements that can be customized, 14
  - Jazz work items, 17–18
  - overview of, 13–14

## D

- Dashboards
  - comparing ClearQuest and Jazz customization areas, 17
  - creating monitoring viewlets, 300
  - in RTC Scrum process, 336–337
- Data
  - accessing records, 114–115
  - attachments, 100–103
  - Back Reference field, 110–113
  - Convert Full\_Name to Login\_Name hook, 122–123
  - corrective actions, 91–92
  - creating parent from child record, 123–126
  - customizing work items, 14
  - defining fields, 35
  - description of work items, 83–86

## Data (*continued*)

- environment information, 87–88
  - external impacts, 89–91
  - hard-coded data in hooks, 272–274
  - HasAttachment hook, 120–121
  - history of work items, 96–98
  - import/export, 163–164
  - importing initial data during deployment, 204–206
  - integration at data-level, 154
  - integration in Jazz-based C/ALM solution, 354
  - internal impacts, 88–89
  - Limit Attachment size hook, 121–122
  - links, 115–116
  - location information, 86–87
  - metrics, 284
  - multiple relationships, 108–109
  - object relations, 106
  - ownership of work items, 99
  - purposes of accumulating, 81–83
  - quality assurance and, 103–104
  - references to unique keys and, 113
  - referencing objects, 114
  - replacing unique keys, 115
  - representation in ClearQuest, 117–118
  - representation in Jazz, 118–119
  - requestor information, 98–99
  - scripts for data representation, 120
  - single relationships, 106–107
  - state-based objects, 105
  - stateless objects, 105–106
  - storing in fields, 7
  - summary, 126
  - test artifacts in RQM, 308
  - test-related, 95–96
  - time-related, 92–94
- Data Hierarchy pattern
- implementing, 233
  - overview of, 39
- Data warehouse, in Jazz repository, 303–304
- Database servers, in infrastructure architecture, 30
- Database vendor tools, for creating test environment, 212–214
- Databases
- change management database, 28–29
  - choosing type in Analysis & Design, 32–33
  - determining number to create, 33

- hook for performing lookup on external, 171
  - moving, 214–216
  - system test and database size, 200
  - test configuration for, 251
  - viewing properties of, 213–214
- Dead End pattern
- design of, 39
  - implementing in ClearQuest, 231
  - implementing in Jazz, 232
- Debugging
- BASIC utilities for, 261
  - using `MsgBox()` function, 255–256
  - using `OutputDebugString()` method, 256–258
  - using tracing information, 258–260
- Decision stage, in workflow item lifecycle, 50–51
- Dedicated actions, in subflow for gathering more information, 70–76
- Default work items, 317–318
- Defects
- change requests due to, 4
  - creating task to fix, 370–371
  - default work items in Jazz, 317
  - key roles in defect and change management, 131
  - quality assurance and, 103
  - reporting, 200, 316
  - test artifacts in RQM, 308–309
  - tracking, 191
- Defect\_Validation hook, 147–149
- Deferring stage, in workflow item lifecycle, 52
- Deleting stage, in workflow item lifecycle, 52
- Demilitarized zones (DMZs), defining infrastructure architecture, 30
- Dependent integrations, ClearQuest, 158–162
- Deployment discipline
- following up on system adoption, 207
  - importing initial data, 204–206
  - installation phase, 202
  - overview of, 201
  - preparation phase, 201
  - setting up environment, 202–203
  - training phase, 206
- Deployment plan, 201
- Description field, for work items, 84
- Descriptive data, 83–86
- Design. *See* Analysis & Design discipline
- Design models, reviewing/signing off on, 40

## Design patterns

- in Analysis & Design, 36–40
  - implementing Child Control pattern, 229–231
  - implementing Closing pattern, 222–225
  - implementing Data Hierarchy pattern, 233
  - implementing Dead End pattern, 231–232
  - implementing Parent Control pattern, 226–229
  - implementing Superuser Modification pattern, 233–234
  - implementing Triage pattern, 224–225
- Designing forms and tabs, 243–245
- Developer Taskboard, in RTC Scrum process, 336
- Developers
- data providing historical information for, 81
  - defining client types, 30
  - testing releases, 250–252
  - work on activities in integrated solution, 368–369

## Development

- being prepared for future requirements, 277
- Child Control pattern, 229–231
- choosing scripting language, 261
- Closing pattern, 222–225
- coding hooks for parallel development, 241
- with common schema (ALM), 222
- comparing/merging schema versions, 245–249
- CQMS (ClearQuest MultiSite) issues, 253–254
- Data Hierarchy pattern, 233
- Dead End pattern, 231–232
- dealing with long selection lists, 265–270
- debugging using `MsgBox()` function, 255–256
- debugging using `OutputDebugString()` method, 256–258
- debugging using tracing information, 258–260
- designing forms and tabs, 243–245
- developer testing, 250–252
- exporting/importing schema portions, 237–238
- hard-coded data and, 272–274
- list of RTC integrations by category, 191
- naming conventions, 262–263
- organizing global scripts by subject, 262
- overview of, 221
- packages, 238–239
- parallel development, 240–241
- Parent Control pattern, 226–229
- promoting release to production, 252–253
- record types in parallel development, 241–243

- releasing versions to production, 250
- resources for, 277–278
- session variables in, 239–240
- storing hooks externally, 249–250
- storing old\_id field for future import, 264
- summary, 278
- Superuser Modification pattern, 233–234
- system testing, 252
- Triage pattern, 224–225
- understanding when stateless record type is required, 261
- unique keys and, 261–262
- updating dynamic lists, 271–272
- Web-related considerations, 274–276
- writing reusable code, 234–237

## Diagrams. *See also* Graphics

- defining and documenting requirements, 25
- entity relationships. *See* ERD (entity relationship diagram)
- in high-level design, 34
- process diagram in change management, 47–48
- state transition, 45–46
- states, 47
- workflows in activity diagrams, 44

## Disciplines

- Analysis & Design. *See* Analysis & Design discipline
- ClearQuest Tool Mentor and. *See* ClearQuest Tool Mentor
- Deployment. *See* Deployment Implementation, 197–199
- Jazz Tool Mentor and, 217–219
- Maintenance, 207–208
- overview of, 197
- Requirements. *See* Requirements discipline
- resources for, 220
- summary, 220
- Testing, 199–200

## DMZs (demilitarized zones), defining infrastructure architecture, 30

## Documentation

- changes affecting, 90
- of corrective actions, 92
- gathering initial, 23
- of requirements, 25–28

## DOORS, RQM integrations, 185

## Duplicating stage, in workflow item lifecycle, 52–53

Dynamic change state, 60–61

Dynamic lists

- adding to choice lists, 273
- updating, 271–272

Dynamic workflow

- moving automatically between states, 57–59
- overview of, 56
- record types in, 56
- single record type having multiple state machines for each issue type, 59–61
- single record type having state machine for each issue, 61–65
- state transition and, 56–57

## E

E-mail notification. *See* Notification

E-mail Reader service, example of ClearQuest integration, 164–167

Eclipse client

- creating process template using, 217–219
- creating project using process template, 217–219
- customizing work items, 17
- import/export tool in, 163–164, 208
- setting permissions for test customization, 321–322

Eclipse Designer

- association by field label, 109
- comparing/merging schema versions, 241, 245–249
- comparing with Windows version of ClearQuest Designer, 18–19
- exporting forms with, 244–245
- supporting parallel development, 241

Editor presentations, Jazz, 2

Effort required

- estimating in requirements gathering, 27–28
- storing in work tasks, 93–94

Electronic signatures

- in access control and security, 289
- in ClearQuest governance, 295–296

Elements, work item

- applets, 12
- attachments, 11
- customizing work items and, 14
- data, 7

links to other work items, 11–12

overview of, 7

pictures or graphics, 12

presentation forms, 7–9

roles, 13

rules, 11

workflows, 9–10

Enhancements

change requests due to, 4

test artifacts in RQM, 309

Entities, defining data fields, 35

Entity relationship diagram. *See* ERD (entity relationship diagram)

Enumeration, Jazz, 3

Environment

creating test environment with ClearQuest tools, 214–216

creating test environment with database vendor tools, 212–214

data related to, 82

fields related to system configuration, 87–88

setting up during deployment, 202–203

ERD (entity relationship diagram)

creating record types in ClearQuest, 198

creating work item types in Jazz, 199

in high-level design, 34

Errors, fixing import. *See also* Defects, 205–206

Estimation of man-hours, in requirements

gathering, 27–28

ETL (extract, transform, load), Insight tool for, 287

Excel, importing record from, 168

Execution states, in RQM test process, 322–323

Export. *See* Import/export

External impacts

applications and, 89

data related to, 82

documentation affected by changes in, 90–91

training materials impacted by changes, 91

users and, 89

Extract, transform, load (ETL), Insight tool for, 287

## F

Features, change requests for adding, 4

Field behaviors, ClearQuest, 3

Field hooks

choice list. *See* Hooks, choice list

default value. *See* Hooks, default value  
 permission. *See* Hooks, permission  
 value changed. *See* Hooks, value changed  
 validation. *See* Hooks, validation

Fields, ClearQuest  
   category fields for organizing information, 320–321  
   compared with Jazz Work item attribute, 2  
   customizing edit permission, 15  
   defining, 35  
   mapping record fields during import, 210  
   permissions, 289, 291–292  
   storing data in, 7  
   storing test data in, 95–96  
   time-related, 94  
   user-defined, 356–358

Fields, making Web forms with dependent fields, 275

Firewalls, 30–32

Forms, ClearQuest  
   compared with Jazz Editor presentations, 2  
   designing, 243–245  
   Presentation forms, 7–9  
   representation of data in ClearQuest, 117–118  
   user-defined fields in, 357

Forms, Jazz. *See* Editor presentations

Forms, making Web forms with dependent fields, 275

Functional requirements, listing, 27

Functional Tester. *See* RFT (Rational Functional Tester)

Functionality packages, 238

## G

GDD (Globally Distributed Development), 253–254

Glossary, in defining and documenting requirements, 25

Governance  
   in ALM, 298  
   in ALM schema, 297  
   in ClearQuest, 290  
   electronic signatures, 295–296  
   monitoring and, 289  
   monitoring in ClearQuest, 297  
   monitoring in RTC, 300

overview of, 287–288

permissions and, 288–289

permissions in ClearQuest, 290–293

permissions in RTC, 299

process control and automation and, 288

process control in ALM, 297

process control in ClearQuest, 290

process control in RTC, 299

reports, 366

resources for, 301–302

roles in ALM, 298

  in RTC, 298

security context in ALM, 297–298

security context in ClearQuest, 294–295

summary, 305

Graphical user interface (GUI), designing forms and tabs and, 243–245

Graphics. *See also* Diagrams  
   defining and documenting requirements, 25  
   in metrics strategy, 280–281  
   presentation insertions, 12

Group box, ClearQuest, 2

Groups, ClearQuest  
   implementing roles with, 131–133  
   query listing members for access control, 144–145

GUI control, ClearQuest, 2

GUI (graphical user interface), designing forms and tabs and, 243–245

## H

Hard-coded data, 272–274

Hardware, location information for, 87

HasAttachment hook, 120–121

Headline field, in work item description, 83–84

Hide data, customizing work items, 15

Histograms, in metrics strategy, 280–281

History  
   data and, 83  
   reasons for saving history of work items, 96–98

History view, in Eclipse Designer, 245–246

Hooks  
   access control, 128, 144–145  
   Action hooks, in ClearQuest, 3  
   automation implemented with, 381–382  
   button hooks in Web development, 274–275



## Hooks (*continued*)

- Choice List hook, 114, 144, 150, 236–238, 270–271
  - cloning hook for creating parent from child record, 123–126
  - code reuse with, 234–237
  - coding hooks for parallel development, 241
  - Commit hook, 230
  - Convert Full\_Name to Login\_Name hook, 122–123
  - default value, 266
  - external storage in Perl not BASIC, 261
  - hard-coded data in, 272–274
  - HasAttachment hook, 120–121
  - initialization, 64, 70–72, 224, 235
  - Limit Attachment size hook, 121–122
  - for lookup on external database, 171
  - notification hook, 58, 60, 62
  - organizing global scripts by subject, 262
  - permission, 293–294, 364
  - storing externally, 249–250
  - validation, 33, 76–78, 147–149, 228
  - value changed, 73–75, 84, 135, 266–267
- HP Mercury Test Director and Quality Center, 176

## I

- IBM Administrator, dependent integration with ClearQuest, 158
- IBM PureCoverage, 157
- IBM Purify, 157
- IBM Quantify, 158
- IBM Rational Build Forge. *See* Build Forge
- IBM Rational ClearCase. *See* ClearCase
- IBM Rational ClearQuest. *See* ClearQuest
- IBM Rational DOORS, 185
- IBM Rational Requirements Composer. *See* Requirements Composer
- IBM RequisitePro. *See* RequisitePro
- IBM RTC. *See* RTC (Rational Team Concert)
- IBM TeamTest, 159
- IBM Tivoli. *See* Tivoli
- IBM Unified Change Management (UCM). *See* UCM (Unified Change Management)
- Icons, in Jazz workflow, 68
- IDEs (Integrated development environments), list of RTC integrations by category, 191
- Idioms. *See* Design patterns, implementing
- Implementation discipline
  - ClearQuest tasks, 198
  - Jazz tasks, 199
  - purpose of, 197–198
- Import/export
  - creating new ClearQuest integrations, 163–164
  - fixing import errors, 205–206
  - importing data during deployment, 204–205
  - importing records with references, 208–210
  - importing updates, 210–211
  - Process Template, 217–219
  - schema portions, 237–238
  - validating imported data, 206
- Import tool, ClearQuest, 205
- Independent integrations, ClearQuest, 156–158
- Individual level metrics, for productivity and efficiency, 280
- Information gathering subflow, 69–76
- Infrastructure, defining infrastructure architecture, 30–32
- Initialization pattern, ClearQuest, 36
- Insight tool
  - for metrics, 287
  - release burndown and velocity charts, 345
- Installation phase, of deployment, 202
- Integrated development environments (IDEs), list of RTC integrations by category, 191
- Integrated solution architecture (ClearCase, ALM, and Build Forge)
  - component relationships in, 368
  - continuous build and validation process, 369–370
  - creating task to fix defect, 370–371
  - developer work on activities in, 368–369
  - overview of, 367
  - periodic releases, 371–372
  - working on test artifacts, 372–373
- Integration, ClearQuest
  - built-in, 156
  - CM API and, 172–175
  - creating new, 162
  - dependent, 158–162
  - E-mail Reader service example, 164–167
  - import/export and, 163–164
  - independent, 156–158
  - OSLC REST API and, 175–176
  - overview of, 155–156

Perl and BASIC API and, 167–172  
 resources for, 195–196  
 summary, 196  
 third-party offerings, 176–177  
 Web services and XML and, 176

Integration, introduction to, 153–155

Integration, Jazz  
 build engines, 190–191  
 Build Forge, 187  
 building new integrations, 192  
 ClearCase Connector, 188  
 ClearQuest Bridge, 180–183  
 ClearQuest Connector, 177–180, 188–190  
 IBM Rational DOORS, 185  
 JIA (Jazz Integration Architecture), 192–193, 355  
 list of RTC integrations by category, 191–192  
 overview of, 177  
 Rational test automation tools, 185–187  
 RequisitePro to RQM, 183–185, 310  
 resources for, 195  
 REST API and, 193–195  
 RQM (Rational Quality Manager) and, 177  
 RTC (Rational Team Concert) and, 187–188  
 STAF (Software Testing Automation Framework), 187  
 summary, 196  
 SVN (Subversion), 190

Integration packages, 238

Internal impacts  
 on components, 88  
 data related to, 82  
 on releases, 88–89

Interviews, in requirements gathering, 23–24

Issues, data in description of, 81

Iterations  
 in ALM schema, 346  
 describing ClearQuest data objects for Agile process, 338  
 planned effort for, 345–346  
 in RTC Scrum process, 332–334

## J

Java APIs, 172–175

Jazz. *See also* RTC (Rational Team Concert)  
 Agile, Realization in RTC, 332–339  
 C/ALM (Collaborative ALM) and, 354–356

comparing ClearQuest and Jazz terminology, 2–3  
 comparing customization elements and terms with ClearQuest, 15–17  
 customizing work items, 17–18  
 databases supported by, 33  
 governance and. *See* RTC (Rational Team Concert) governance  
 implementation tasks, 199  
 integrations. *See* Integration, Jazz  
 monitoring with Jazz, 302  
 process control with Jazz, 301  
 permission with Jazz, 301  
 report resources, 302–303  
 representation of data in, 118–119  
 roles in, 142–143  
 setting up customizations during deployment, 204  
 setting up environment during deployment, 202–203  
 types of links to Jazz work items, 115–116  
 workflow, 66–68

Jazz Foundation Services, in C/ALM solution, 355

Jazz Integration Architecture (JIA)  
 building new Jazz integrations, 192–193  
 in Jazz-based C/ALM solution, 355

Jazz Tool Mentor  
 creating project using process template, 217–219  
 overview of, 217

JIA (Jazz Integration Architecture)  
 building new Jazz integrations, 192–193  
 in Jazz-based C/ALM solution, 355

## L

License servers, defining infrastructure architecture, 30–31

Limit Attachment size hook, 121–122

Links  
 to attachment files (CQ), 101–102  
 types of (Jazz), 115–116  
 to work items, 11–12

List View, Back Reference field and, 110

Lists  
 choice lists. *See* Choice list, Clear Quest  
 dynamic lists. *See* Dynamic lists

Location information  
   in describing physical artifacts involved in  
   change, 86–87  
   locating items/components that must be  
   changed, 82  
 Login name, converting full name to, 122–123  
 Lookups, hook for performing lookup on external  
 database, 171

## M

Maintaining requirements, 28–29  
 Maintenance discipline  
   defining change process, 207–208  
   improving maintainability, 208  
   ongoing support, 208  
   overview of, 207  
 Manage Section wizard, 318  
 Management (decision makers)  
   benefits of integration to, 154  
   data providing information for managers, 81  
   getting agreement regarding requirements, 28  
 Manual Tester, RQM replacing, 307  
 Mastership, in multisite environment  
   addressing changes to, 253–254  
   testing, 254  
 MCIF (Measured Capability Improvement  
 Framework), 287, 301  
 Members, 311–312  
 Metrics  
   BIRT (Business Intelligence and Reporting  
   Tools), 286  
   categorizing by levels, 280  
   ClearQuest, creating Agile schema in, 345  
   Crystal reports, 285–286  
   data supporting, 284  
   Insight tool for, 287  
   overview of, 279–280  
   Publishing Engine for document generation,  
   287  
   Report Server for managing ClearQuest  
   reports, 286–287  
   resources for, 301–302  
   strategy for, 280–284  
   summary, 305  
   tools for, 284–285  
   units for time-related data, 94

Microsoft Visual Source Safe, integration with  
 ClearQuest, 161  
 Modeling. *See also* UML (Unified Modeling  
 Language)  
   list of RTC integrations by category, 191  
   promotion process, 379–382  
 Monitoring  
   in ClearQuest, 297  
   in governance generally, 289  
   in Jazz, 300  
 MsgBox ( ) function, debugging with, 255–256  
 Multiple relationships, object relations in  
 ClearQuest, 106, 108–109  
 Multiple roles  
   implementing implicitly, 134  
   overview of, 130  
   setting choices based on, 146–147

## N

Naming conventions, in schema development,  
 262–263  
 New session, notification hook, 58  
 Notification pattern, 36  
 Notifications  
   Assign action and, 58  
   customizing work items, 16  
   in debugging, 256  
   roles in, 127  
   rules governing, 11  
   setting environment, 202  
   SLAs and, 365–366  
 Not\_Resolved state type, 54

## O

Objects  
   relations. *See* Relationships  
   state-based, 105  
   stateless, 104–106  
   test artifacts, 308–309  
 old\_id field, storing for future reference, 264  
 Ongoing support, in maintenance discipline, 208  
 Open Service for Lifecycle Collaboration.  
   *See* OSLC (Open Service for Lifecycle  
   Collaboration)  
 Open Unified Process (OpenUP), 45

- OpenUP (Open Unified Process), 45
- Operating systems (OSs), environment information and, 87–88
- Operation behavior or extensions, Jazz, 3
- Organization level metrics, for productivity and efficiency, 280
- OSLC (Open Service for Lifecycle Collaboration)
  - building new Jazz integrations, 193–195
  - in Jazz-based C/ALM solution, 355
  - REST API compliance with, 175–176
- OSs (operating systems), environment information and, 87–88
- `OutputDebugString()` method, debugging with, 256–258
- Ownership
  - assigning solution providers in process of taking corrective actions, 91
  - data, 99
  - defined, 130
  - misassignment of, 128–129
  - unassigned changes and, 128

## P

- Packages (in ClearQuest)
  - creating, 239
  - enabling for editing, 239
  - overview of, 238–239
  - types of, 238
- `packageutil`, creating packages with, 239
- Parallel development
  - coding hooks for, 241
  - designing forms and tabs, 243–245
  - overview of, 240–241
  - record types in, 241–243
  - storing hooks externally and, 249–250
- Parent/Child control, 110–112
- Parent Control pattern
  - design of, 38
  - implementing, 226–229
- Parent records, creating from child record, 123–126
- Performance benchmarking, by Web administrators, 200
- Performance Tester. *See* RPT (Rational Performance Tester)
- Perl
  - choosing scripting language and, 261

- Convert Full\_Name to Login\_Name hook, 122–123
- creating new ClearQuest integrations, 167–172
- HasAttachment hook, 120–121
- Limit Attachment size hook, 121–122
- storing hooks externally, 249–250
- use of session variables, 240
- Permissions
  - in ClearQuest, 290–293
  - in governance, 288–289
  - in Jazz, 299
  - Jazz permissions compared with ClearQuest
    - Access control, 3
    - roles and, 13, 142–143
    - for test customization, 321–322
    - workflow rules, 46
- Petri nets, modeling workflows with, 43
- Pictures, as presentation insert, 12
- Plans, Jazz, 2
- Postponing stage, in workflow item lifecycle, 52
- Preconditions, Jazz, 3
- Preparation phase, of deployment, 201
- Presentation
  - customizing work items, 15
  - Jazz Presentation compared with ClearQuest
    - GUI control, 2
    - representation of data, 118
- Presentation forms, 7–9
- Prioritization
  - of corrective actions, 91
  - in requirements gathering, 27
- Process Configuration, customizing work items, 317
- Process control
  - in ALM schema, 297
  - in ClearQuest, 290
  - in governance, 288
  - in Jazz, 299
- Process entities, in promotion request system, 377–378
- Process lifecycle
  - managing release promotion and, 374–375
  - promotion requests and, 379
- Process Template, Jazz
  - compared with ClearQuest Schema, 2
  - creating new project area with imported template, 219

Process Template, Jazz (*continued*)

- exporting, 217–218
- generating, 217
- importing, 218–219

Processes

- components in promotion process, 376–379
- integration at process-level, 154
- representation in workflows, 45–49
- rules in promotion process model, 379–381
- WBM (WebSphere Business Modeler), 47–48

Product backlog

- describing ClearQuest data objects for Agile process, 338
- ranking Jazz stories in, 330–331
- showing Jazz iterations in, 331–332

Project level metrics, for productivity and efficiency, 280

Project management, list of RTC integrations by category, 192

Projects

- classification of project-related work items, 6–7
- work projects in ClearCase, 368–369

Promotion process

- model for promotion process, 379–382
- overview of, 374
- process components, 376–379
- solutions, 375
- status assessment, 374–375

Promotion work items, 6

Publishing Engine, for document generation, 287

PureCoverage, integration with ClearQuest, 157

Purify, integration with ClearQuest, 157

## Q

Quality assurance

- benefits of integration to, 154
- data related to, 83, 103–104

Quality indicators, in reports, 284

Quality management, list of RTC integrations by category, 192

Quality Manager. *See* RQM (Rational Quality Manager)

Quantify, integration with ClearQuest, 158

Queries, comparing ClearQuest and Jazz

- customization areas, 17

Query wizard, 285

Questionnaires, in requirements gathering, 24–25

Quick Information section, Jazz, 2

## R

R (ready) state type, UCM, 54

Ranking stories, in RTC Scrum process, 330–331

Rational AppScan, integration of RQM and RTLM with, 185–187

Rational Build Forge. *See* Build Forge

Rational ClearCase. *See* ClearCase

Rational ClearQuest. *See* ClearQuest

Rational CM API. *See* CM (Configuration Management) API

Rational DOORS, 185

Rational Functional Tester (RFT)

- automating test scripts with, 315
- integration of RQM and RTLM with, 185–187

Rational Insight tool. *See* Insight tool

Rational Performance Tester (RPT)

- integration of RQM and RTLM with, 185–187
- test simulation with, 200

Rational Publishing Engine, for document generation, 287

Rational Quality Manager. *See* RQM (Rational Quality Manager)

Rational Requirements Composer (RRC). *See* Requirements Composer

Rational RequisitePro. *See* RequisitePro

Rational Team Concert Eclipse client. *See* Eclipse client

Rational Team Concert (RTC). *See* RTC (Rational Team Concert)

Rational test automation tools, 185–187

Rational Test Lab Manager. *See* RTLM (Rational Test Lab Manager)

Rational Unified Process (RUP). *See* RUP (Rational Unified Process)

Rational University, 206

Reassign action, 70

Recalculate Choice list property, 265–267

Record forms, representation of data in ClearQuest, 117

Record type family, ClearQuest, 2

Record types, ClearQuest

- compared with Jazz Work item type, 2
- creating parent record from child record, 123–126

- customizing, 18–19
  - in dynamic workflow, 56
  - in parallel development, 241–243
  - single record type having multiple state machines for each issue type, 59–61
  - single record type having state machine for each issue, 61–65
  - stateful, 56
  - stateless, 104–106
  - user-defined roles and, 137–138
  - using roles stateless record type with static roles, 136–137
- Records**
- accessing via referencing, 114–115
  - importing updates to existing, 210–211
  - importing with references, 208–210
  - mapping fields during import, 210
  - with more than one field as unique key, 261–262
  - permissions for access control and security, 289
  - Security Context and, 293
  - test artifacts in RQM, 308
- Reference\_List**
- Back Reference field and, 110–113
  - types of object relations in ClearQuest, 106, 108–109
- References**
- importing records with references, 208–210
  - to objects, 114
  - pointing to unique keys, 113
  - types of object relations in ClearQuest, 106–107
- Rejecting stage, in workflow item lifecycle, 52**
- Relationships**
- accessing records, 114–115
  - Back Reference field in object relations, 110–113
  - links, 115–116
  - multiple, 108–109
  - overview of, 106
  - references to unique keys and, 113
  - referencing objects, 114
  - replacing unique keys, 115
  - single, 106–107
  - between test artifacts, 308–309
- Release work items, 6**
- Releases**
- burndown and velocity charts, 345
  - developer testing, 250–252
  - internal impacts on, 88–89
  - managing promotion process, 374
  - model for promotion process, 379–382
  - overview of, 250
  - periodic, 371–372
  - process components, 376–379
  - promoting to production, 252–253
  - solutions, 375
  - status assessment, 374–375
  - system testing, 252
- Reloading records, notification hook for, 58**
- Reopening stage, in workflow item lifecycle, 51–52**
- Report Server, for managing ClearQuest reports, 286–287**
- Reports**
- comparing ClearQuest and Jazz customization areas, 17
  - defects, 316
  - in metrics strategy, 281
  - test defects, 316
- Representation of data**
- in ClearQuest, 117–118
  - in Jazz, 118–119
  - scripts in ClearQuest, 120
- Request record type**
- in ALM schema, 346, 348–349
  - describing data objects for Agile process, 338
  - suggested fields for, 339–340
  - workflows of, 343–344
- Requestor information, 98–99**
- Requirements Composer (RRC)**
- documenting requirements, 25–26
- Requirements discipline**
- Agile practices applied to, 28
  - analyzing requirements in establishing test strategy, 310
  - default work items in Jazz, 317
  - defining and documenting, 25–28
  - gathering initial documentation, 23
  - getting agreement regarding, 28
  - interviews for gathering, 23–24
  - maintaining, 28–29
  - overview of, 23
  - questionnaires for gathering, 24–25
  - resources, 40

## Requirements discipline (*continued*)

- section of test plan, 311
- summary, 40–41
- test artifacts in RQM, 308–309
- verifying test case coverage of, 315

## Requirements management, list of RTC

- integrations by category, 192

## RequisitePro

- analyzing requirements in establishing test strategy, 310
- dependent integration with ClearQuest, 159
- documenting requirements, 25
- managing release promotion and, 375
- RQM integrations, 183–185

## Reserved names, in schema development, 263

## Resolution pattern, 40

## Resolution stage, in workflow item lifecycle, 51

## Resolution states, 53–55

## Resolutions

- effort required for, 93
- in Jazz workflow, 67

## Resolved state type, 54

## Resources

- client interactions with resources on servers, 173
- enhancements in Jazz version 2.0, 19

## Responsible

- defined, 130
- setting based on role object, 147–149
- setting based on Single role, 145
- unassigned changes and, 128

## REST API

- building new Jazz integrations, 193–195
- creating new ClearQuest integrations, 175–176

## Results, test artifacts in RQM, 308

## Reusable assets

- code reuse, 105, 234–237
- exporting/importing schema portions, 237–238
- list of RTC integrations by category, 191
- overview of, 234
- packages, 238–239

## Review, of test plan, 313

## RFT (Rational Functional Tester)

- automating test scripts with, 315
- integration of RQM and RTLM with, 185–187

## Risk work items, 6

## Role-based access control, 379

## Roles

- in access control hook, 144–145
- in ALM governance, 298
- in ALM schema, 140–141
- areas and groups and, 131
- auto-assignment of, 145–149, 273–274
- change request process and, 131
- comparing Agile and ALM schema, 346–347
- customizing work items, 16
- defined, 130
- implementing implicitly, 133–136
- implementing with groups, 132–133
- issues/typical problems addressed by, 128–129
- in Jazz, 142–143
- in Jazz governance, 299
- organizing questionnaires by, 25
- overview of, 127
- in process model, 376–377
- in Scrum process, 334–335
- security and, 139
- stateless record type and, 136–137
- summary, 151
- terminology related to and types of, 130
- user-defined, 137–139, 150
- users activities and, 13
- workflow rules, 46

## RPT (Rational Performance Tester)

- integration of RQM and RTLM with, 185–187
- test simulation with, 200

## RQM (Rational Quality Manager)

- Build Forge integration with, 187
- ClearQuest Bridge, 180–183
- ClearQuest Connector, 177–180
- combining with DOORs in Jazz integrations, 185
- combining with RequisitePro in Jazz integrations, 183–185
- creating dashboard viewlets for monitoring, 300
- customizing work items, 17
- databases supported by, 33
- integration at data-level, 154
- integration with ClearQuest, 163, 180–183
- integration with Rational test automation tools, 185–187
- integration with STAF, 187
- integrations, 177

- roles in, 142
  - setting up Jazz environment during deployment, 203
  - test entities and relationships, 307–310
  - test plan and test case objects in, 6
  - testing work items. *See* Testing work items, in RQM
  - what it is, 307
- RTC (Rational Team Concert)
- as ALM tool, 7
  - Build engines, 190–191
  - ClearCase Connector, 188
  - ClearQuest integration with, 188–190
  - customizing work items, 17
  - databases supported by, 33
  - dependent integration with ClearQuest, 163
  - implementing Closing pattern, 224–225
  - integrations, 187–188
  - Jazz integration with ClearQuest, 180–183
  - listing integrations by category, 191–192
  - representation of data in Jazz, 118–119
  - roles in, 142
  - setting permissions for test customization, 321–322
  - setting up Jazz environment during deployment, 203
  - support for customization from Web clients, 119
  - SVN (Subversion) used for source control, 190
- RTC (Rational Team Concert) governance
- monitoring in, 300
  - overview of, 298
  - permissions in, 299
  - process control in, 299
- RTC (Rational Team Concert), Scrum process in
- dashboards, 336–337
  - Developer Taskboard, 336
  - iterations and sprint planning, 332–334
  - Scrum roles, 334–335
  - setting up multiple teams for multiple releases, 331–332
  - stories, 330–331
- RTLM (Rational Test Lab Manager)
- based on RQM, 307
  - built-in integrations, 177
  - integration at tools-level, 154
- Rules
- customizing work items, 16

- governing change process, 11
  - workflow definition, 46
- RUP (Rational Unified Process)
- applying to change management, 21–22
  - definition of roles, 127
  - workflows in software development, 44–45

## S

- SCCB (Software Change Control Board), 131
- Scenarios, in Jazz-based C/ALM solution, 355
- Schedules, test artifacts in RQM, 308
- Schema packages, 238
- Schemas
- ClearQuest schema compared with Jazz Process Template, 2
  - comparing/merging schema versions, 245–249
  - componentizing, 241–243
  - exporting/importing schema portions, 237–238
  - high-level design in ClearQuest, 33–34
  - upgrading, 253
  - use in development, 222
- Scripts
- choosing scripting language, 261
  - cloning hook for creating parent from child record, 123–126
  - Convert Full\_Name to Login\_Name hook, 122–123
  - for data representation, 120
  - for executing tests in RQM, 315–316
  - function Defect\_AccessControl, 132
  - function Defect\_Validation, 145, 147
  - function GetLoginfromFullName, 123
  - function SetRequiredness, 362
  - HasAttachment hook, 120–121
  - import script, 169–171
  - Limit Attachment size hook, 121–122
  - organizing global scripts by subject, 262
  - sub allowedactions\_ChoiceList, 150
  - sub ALMActivity\_Validation, 77
  - sub AssignApprover, 78
  - sub attachments\_ValueChanged, 120
  - sub Bug\_Initialization, 223
  - sub ChildrenStatus, 228
  - sub ChoiceList, 236
  - sub Comp\_ChoiceList, 269
  - sub comp\_level2\_ChoiceList, 270



Scripts (*continued*)

- sub Defect\_Commit, 229
- sub Defect\_Initialization, 234
- sub Defect\_Notification, 58
- sub Defect\_Validation, 227
- sub EnhancementRequest\_Initialization, 71–72
- sub feature\_CloneParentCR, 124
- sub getEntityURL, 256
- sub GetLoginfromFullName, 122
- sub GetNextRole, 273
- sub Issue\_Initialization, 64
- sub Issue\_Notification, 60, 62
- sub keywords\_ValueChanged, 271
- sub Limit\_Attachment, 121
- sub note\_entry\_ValueChanged, 75
- sub Populate\_list, 360
- sub project\_ValueChanged, 266
- sub resolution\_ChoiceList, 273
- sub responsible\_ChoiceList, 146
- sub SetLog, 73
- sub ucm\_view\_Permission, 292
- sub userfieldvalue\_2\_ChoiceList, 267
- test artifacts in RQM, 308

## Scrum

- dashboards, 336–337
- Developer Taskboard, 336
- framework for Agile development, 326–329
- iterations and sprint planning, 332–334
- Realization with ClearQuest, 339–348
- roles, 334–335
- setting up multiple teams for multiple releases, 331–332
- stories, 330–331

## Section, Jazz

- compared with ClearQuest Group box, 2
- customizing test work items, 318–320
- representation of data and, 118
- setting permissions for test customization, 321–322

## Security

- Access\_Control hook in ClearQuest, 128
- defining infrastructure architecture, 31
- governance and, 288–289
- roles and, 139

## Security Context feature, ClearQuest

- in ALM schema, 297–298

in ClearQuest governance, 293–295

overview of, 139

records and, 293

Security testing, list of RTC integrations, 192

Selection lists. *See* Choice list, Clear Quest

Sequence diagrams, workflows in, 44

## Servers

client interactions with resources on, 173

defining client types, 30–31

installing during deployment, 202

for managing ClearQuest reports, 286–287

Service Level Agreements. *See* SLAs (Service Level Agreements)

Service Request Manager (SRM), 154, 172

Session variables, 239–240

Single relationships, types of object relations in ClearQuest, 106–107

## Single roles

implementing implicitly, 134

overview of, 130

setting Responsible based on, 145–146

## SLAs (Service Level Agreements)

activating rules, 365

background and examples, 363–364

defining rules, 364–365

external program for, 366–367

governance reports, 366

notifications, 365–366

overview of, 363

resources for, 383

## SME (subject matter expert), 208

## Software

internal impacts on components, 88

location information, 87

## Software Change Control Board (SCCB), 131

## Software development processes

OpenUP (Open Unified Process), 45

RUP (Rational Unified Process), 44–45

## Software Testing Automation Framework (STAF), 187

## Solution providers

assigning in process of taking corrective actions, 91

data providing information for, 81

key roles in defect and change management, 131

## Solutions, in promotion process, 375

- Sprint record type
  - suggested fields for, 340–341
  - workflows of, 344–345
- Sprints
  - in ALM schema, 348–349
  - describing ClearQuest data objects for Agile process, 338
  - in Scrum process, 332–334
- SQL query
  - listing group members for access control, 144–145
  - listing in SLA, 368–369
- SRM (Service Request Manager), 154, 172
- STAF (Software Testing Automation Framework), 187
- Stages, of workflow, 50–53
- Stakeholders
  - getting agreement regarding requirements, 28
  - interviews for gathering requirements from, 23–24
  - key roles in defect and change management, 131
  - questionnaires for gathering requirements from, 24–25
- State-based objects, 105
- State group, Jazz, 2
- State pattern, 36
- State transition
  - diagramming, 45–46
  - in dynamic workflow, 56–57
  - moving automatically between states, 57–59
  - test artifact state transition constraints, 312
- State transition matrix, ClearQuest
  - compared with Jazz Workflow, 2
  - creating, 49
  - workflow design with, 9
- State transition tables, 9
- State type, ClearQuest
  - compared with Jazz State group, 2
  - workflow and, 53–56
- Stateful record types
  - ALM Security Context and, 297
  - overview of, 56
- Stateless objects, 105–106
- Stateless record types
  - ALM schema roles, 140
  - ALM Security Context and, 297
  - importing, 205
  - overview of, 104–106
  - understanding when they are required, 261
  - user-defined roles and, 137–139
  - using roles stateless record type with static roles, 136–137
- States
  - auto-change, 58–59
  - comparing ClearQuest state with Jazz state, 2
  - customizing test execution states, 322–323
  - diagramming, 47
  - dynamic change state, 60–61
  - in Jazz workflow, 66–67
  - workflow rules, 46
- Static roles, 136
- Status assessment, in promotion process, 374–375
- Steps to Reproduce field, in work item description, 85
- Stories, in Scrum process, 330–331
- Storyboards
  - defining and documenting requirements, 26
  - for functional requirements, 27
- Subflows
  - for build approval, 76–78
  - deviation needed to gather more information, 69–76
  - overview of, 68–69
- Subject matter expert (SME), 208
- Submission stage, in workflow item lifecycle, 50
- Submit forms, representation of data in
  - ClearQuest, 117
- Subversion (SVN), 190
- Superuser Modification pattern
  - design of, 39–40
  - implementing, 233–234
- Supplementary specifications, in requirements gathering, 27
- Support, in maintenance discipline, 208
- SVN (Subversion), 190
- System
  - access permissions in ClearQuest, 290–291
  - defining scope of in requirements gathering, 26–27
  - following up on adoption, 207
  - tests, 200, 252

## T

### Tabs

- comparing ClearQuest with Jazz, 2
- designing, 243–245
- representation of data in Jazz, 118

Task-Quality, default work items in Jazz, 317

Task record type, in ALM schema, 346, 348

Task-Review, default work items in Jazz, 317

### Tasks

- activity subclasses, 5
- assigning members to, 311–312
- creating task to fix defect, 370–371
- default work items in Jazz, 317

Team Concert. *See* RTC (Rational Team Concert)

Team level metrics, for productivity and efficiency, 280

### Teams

- describing ClearQuest data objects for Agile process, 338
- in Scrum process, 331–332

TeamTest, integration with ClearQuest, 159

Telelogic Harmony, 44–45

Termination dates, for actions, 92–93

Test Case field, in work item description, 85–86

### Test cases

- associating with requirements and verifying coverage, 315
- categories for organizing information in, 320–321
- creating and associating with test plan, 314
- creating scripts for executing, 315–316
- creating suite of, 316
- customizing test work items, 316
- test artifacts in RQM, 308–309

Test elements, in work item classification, 6

### Test environment

- creating with ClearQuest tools, 214–216
- creating with database vendor tools, 212–214
- promoting release to production, 252–253
- storing hooks externally, 250

Test Lab Manager. *See* RTLM (Rational Test Lab Manager)

### Test plan

- assigning members to tasks, 311–312
- categories for organizing information in, 320–321
- creating, 311

creating test cases and associating with test plan, 314

customizing test work items, 316

reviewing, 313

test artifacts in RQM, 308–309

### Test suite

creating, 316

test artifacts in RQM, 308–309

### Testers

- data providing historical information for, 81
- in test process, 312–318

Testing stage, in workflow item lifecycle, 51

### Testing work items, in RQM

- analyzing requirements and establishing test strategy, 310
- assigning members to test plan tasks, 311–312
- creating scripts for executing test cases, 315–316
- creating suite of test cases, 316
- creating test cases and associating with test plan, 314
- creating test plan, 311
- customizing test execution states, 322–323
- customizing test work items, 316–317
- default work items, 317–318
- executing tests, 316
- organizing test information using categories, 320–321
- reporting defects found during testing, 316
- reviewing test plan, 313
- setting permissions for test customization, 321–322
- summary, 324
- testing specific work items, 318–320
- verifying test case coverage of requirements, 315

TestManager, RQM replacing, 307

### Tests/testing

- data related to, 82–83
- developers in, 250–252
- effort required for, 93
- fields for test data, 95–96
- integration of RQM and RTLM with test automation tools, 185–187
- key roles in defect and change management, 131
- overview of Testing discipline, 199–200
- system testing, 252

- working on test artifacts for integrated solution, 372–373
- Third-party integrations, with ClearQuest, 176–177
- Time-related data
  - amount of effort required, 93–94
  - overview of, 82
  - termination dates, 92–93
- Tivoli
  - SRM (Service Request Manager), 154, 172
  - TPM (Tivoli Provisioning Manager), 163
- Tools, for metrics
  - BIRT (Business Intelligence and Reporting Tools), 286
  - Crystal reports, 285–286
  - overview of, 284–285
  - rational Insight tool for, 287
  - Rational Publishing Engine for document generation, 287
  - Report Server for managing ClearQuest reports, 286–287
- Tools, integration at tools-level, 154
- TPM (Tivoli Provisioning Manager), 163
- Tracing information, debugging with, 258–260
- Training materials, external impacts on, 91
- Training phase, in deployment discipline, 206
- Transitions. *See also* State transition, 67
- Transparency, in Jazz-based C/ALM solution, 354
- Triage pattern
  - design of, 37–38
  - implementing in ClearQuest, 224–225
  - implementing in Jazz, 225
- TSRM (Tivoli Service Request Manager). *See* SRM (Service Request Manager)

## U

- UCM (Unified Change Management)
  - integration with ClearQuest, 160–161
  - periodic releases, 371–372
  - state types and, 53–55
  - work projects in ClearCase, 368–369
  - working on test artifacts, 372–373
- UML (Unified Modeling Language)
  - in high-level design, 34
  - modeling workflows with, 43
  - workflow design with, 9

- workflows in, 44
- Unified Process. *See* RUP (Rational Unified Process)
- Unique keys
  - dealing with records with more than one field as unique key, 261–262
  - references to, 113–115
  - replacing, 115
- Units of work
  - measuring time-related data, 94
  - time measurements, 284
- Updates
  - dynamic lists, 271–272
  - importing, 210–211
- User acceptance testing, in release process, 252
- User-defined fields
  - in ClearQuest solution, 356–358
  - defining requiredness, 361–362
- User-defined roles
  - ChoiceList hook for, 150
  - overview of, 137–139
- User interface
  - defining, 36
  - designing forms and tabs, 243–245
- Users
  - benefits of integration to, 154
  - business rules, 378–379
  - defining client types, 30
  - external impacts and, 89
  - following up on system adoption, 207
  - getting agreement regarding requirements, 28
  - ongoing support, 208
  - permissions for access control and security, 288–289
  - roles, 13
  - roles in different areas, 131
  - training, 206
- User's privileges, ClearQuest, 3

## V

- Validation
  - continuous build and validation process, 369–370
  - hook. *See* Hooks, validation
  - of imported data, 206

Velocity charts, releases and, 345  
 Version control, list of RTC integrations by category, 192  
 VersionOne, ClearQuest integration with, 177  
 Visual Source Safe, ClearQuest integration with, 161  
 Vocabulary control, glossary in defining and documenting requirements, 25

## W

W (waiting) state type, UCM, 54  
 WBM (WebSphere Business Modeler), 47–48  
 Web administrators, performance benchmarking by, 200  
 Web clients  
   limitations of, 276  
   support for customization from, 119  
   testing releases with, 252  
 Web-related development considerations  
   enabling button hooks, 274–275  
   limitations of Web clients, 276  
   making forms with dependent fields, 275  
   overview of, 274  
 Web servers, defining infrastructure architecture, 30  
 Web services, ClearQuest integration with, 176  
 WebSphere Business Modeler (WBM), 47–48  
 Windows version, of ClearQuest Designer, 18–19, 213  
 Work item attribute, Jazz, 2, 7  
 Work item type category, Jazz, 2  
 Work item type, Jazz, 2  
 Work items  
   classification of. *See* Classification of work items  
   comparing ClearQuest and Jazz terminology, 2–3  
   customizing. *See* Customizing work items  
   defined, 1  
   elements. *See* Elements, work item  
   summary, 20  
   in test process. *See* Testing work items, in RQM  
   what's new in 2.0 work items, 19  
 Workflows  
   in ALM schema, 65–66  
   in clearQuest, creating Agile schema in, 343–345  
   customizing work items, 15  
   defining, 35  
   dynamic. *See* Dynamic workflow  
   Jazz, 66–68  
   Jazz workflow compared with ClearQuest State transition matrix, 2  
   overview of, 9–10, 43  
   process representation, 45–49  
   software development processes, 44–45  
   stages in, 50–53  
   state types, 53–56  
   subflows. *See* Subflows  
   summary, 78–79  
 Workspace Versioning and Configuration Management (WVCM), 172–173  
 WVCM (Workspace Versioning and Configuration Management), 172–173

## X

XML, ClearQuest integration with, 176

*This page intentionally left blank*



# This could be the best advice you get all day

The IBM® International Technical Support Organization (ITSO) develops and delivers high-quality technical materials and education for IT and business professionals.

These value-add deliverables are IBM Redbooks® publications, Redpapers™ and workshops that can help you implement and use IBM products and solutions on today's leading platforms and operating environments.

***See a sample of what we have to offer***

***Get free downloads***

***See how easy it is ...***

**[ibm.com/redbooks](http://ibm.com/redbooks)**

- Select from hundreds of technical deliverables
- Purchase bound hardcopy Redbooks publications
- Sign up for our workshops
- Keep informed by subscribing to our weekly newsletter
- See how *you* can become a published author

We can also develop deliverables for your business. To find out how we can work together, send a note today to: [redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)