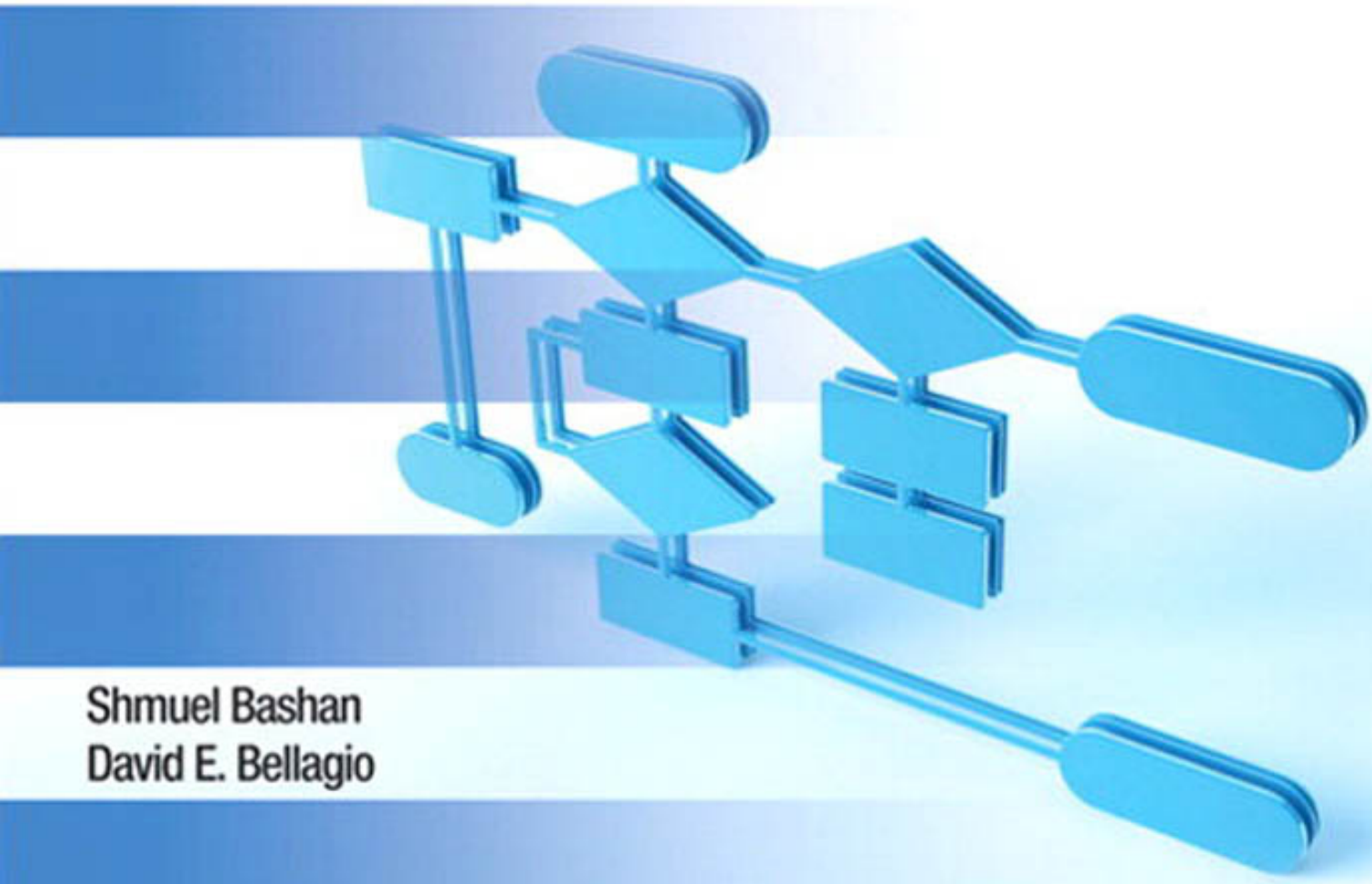
IBM PRESS

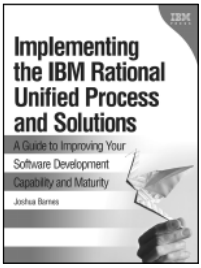
Work Item Management with IBM Rational ClearQuest and Jazz

A Customization Guide



Shmuel Bashan
David E. Bellagio

Related Books of Interest

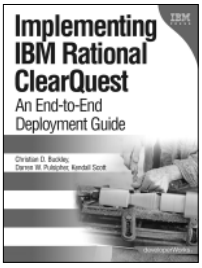


Implementing the IBM® Rational Unified Process® and Solutions

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®

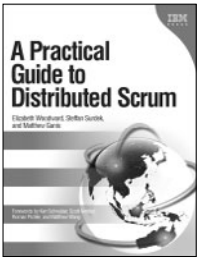
An End-to-End Deployment Guide

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.

Related Books of Interest



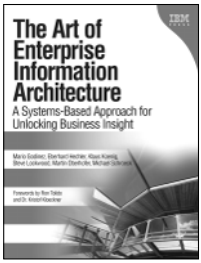
A Practical Guide to Distributed Scrum

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

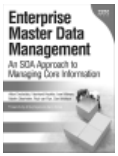
Software Test Engineering with IBM Rational Functional Tester

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

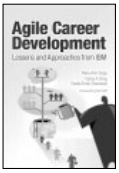
ISBN-10: 0-13-236625-8

The Business of IT

How to Improve Service and Lower Costs

Robert Ryan, Tim Raducha-Grace

ISBN-10: 0-13-700061-8

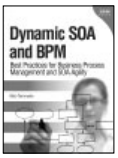


Agile Career Development

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

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

IBM PRESS

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

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

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

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

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).

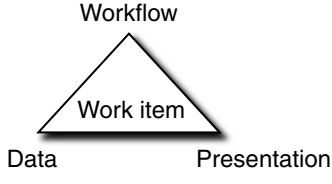


Figure P.1 Elements of the work item (the work item triangle): workflow, data, and presentation

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 mange 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.

CHAPTER 11

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

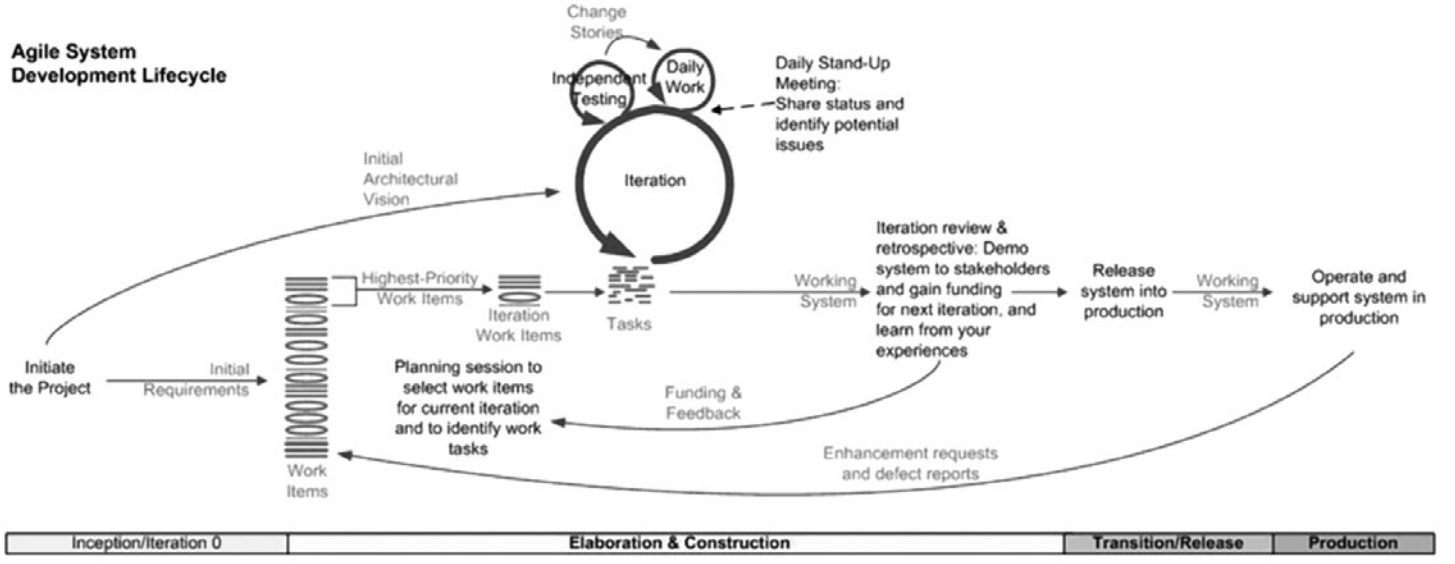


Figure 11.1 The Agile System Development Lifecycle (copyright Scott W. Ambler)

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.

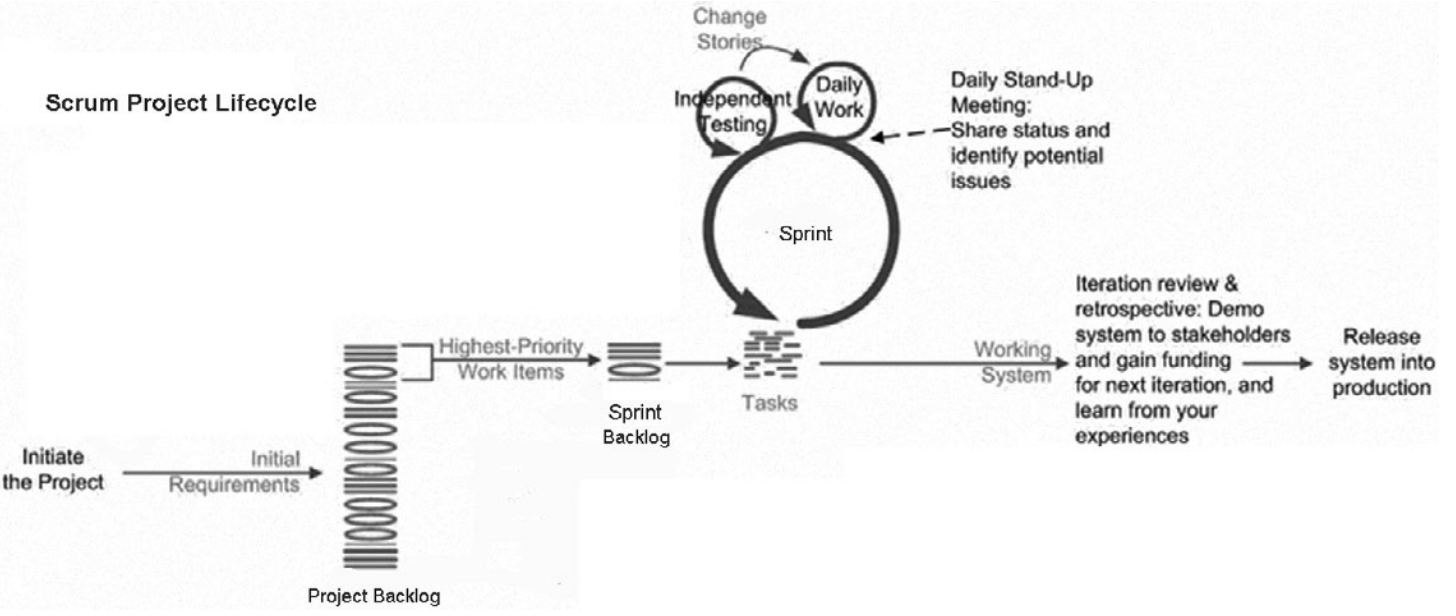


Figure 11.2 The Scrum Project Lifecycle (copyright Scott W. Ambler)

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.

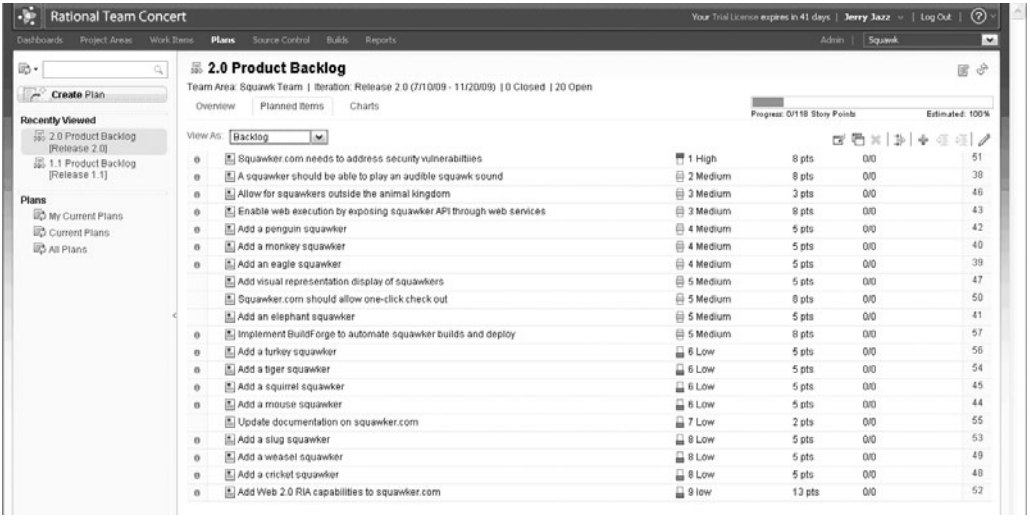


Figure 11.3 The product backlog of Release 2.0

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.

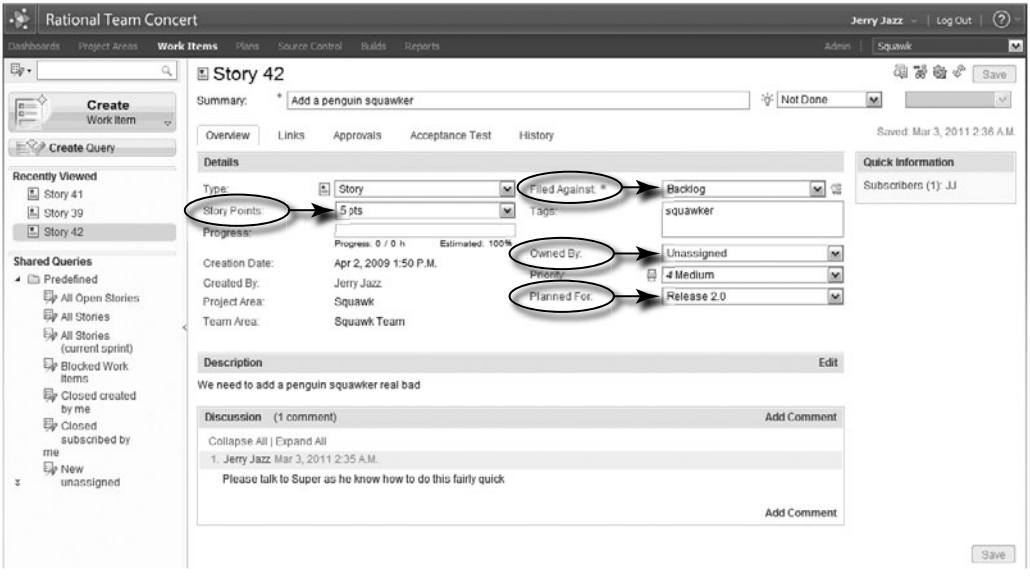


Figure 11.4 The Story work item

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).

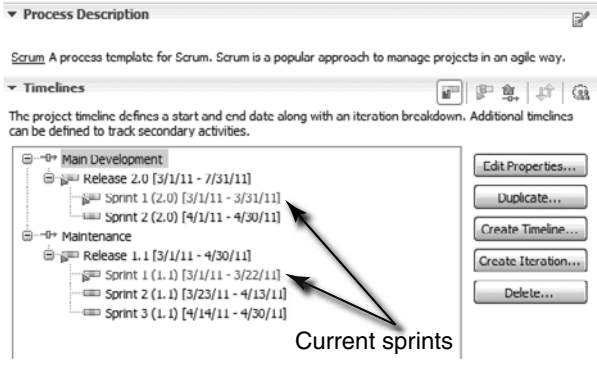


Figure 11.6 Sprints of different releases

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.

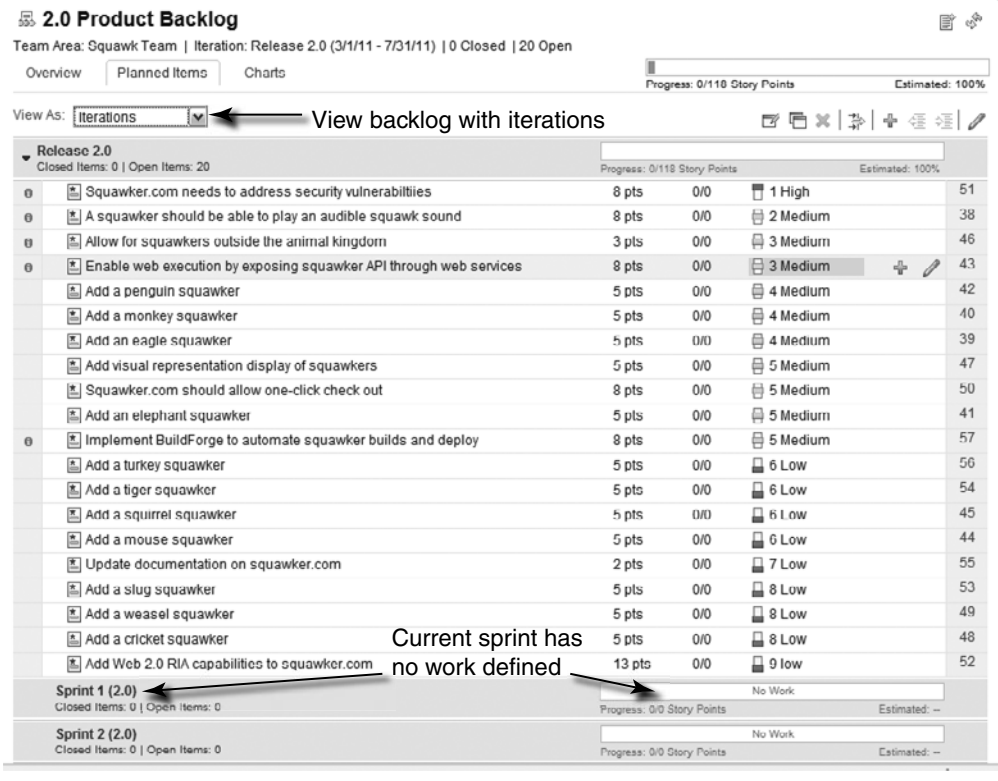


Figure 11.7 Release 2.0 backlog showing iterations

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.

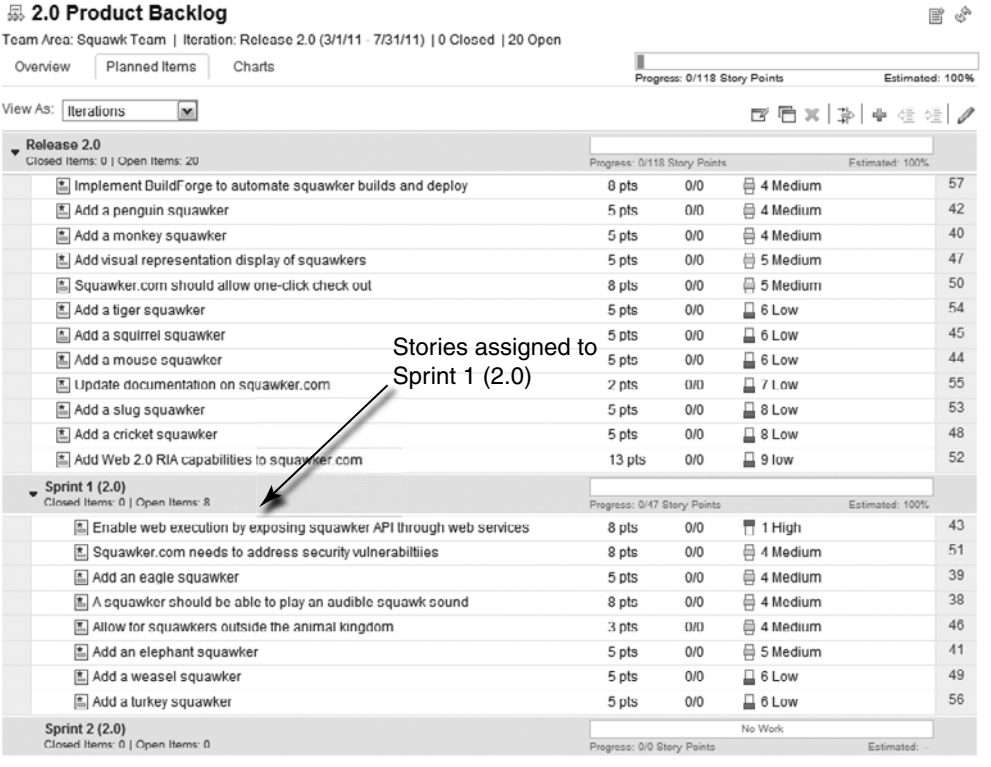


Figure 11.8 Assigning stories from the product backlog to a sprint

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.

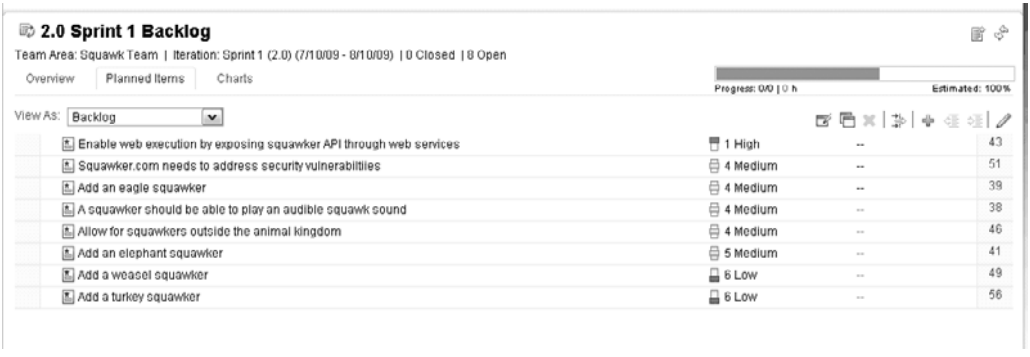


Figure 11.9 The sprint backlog

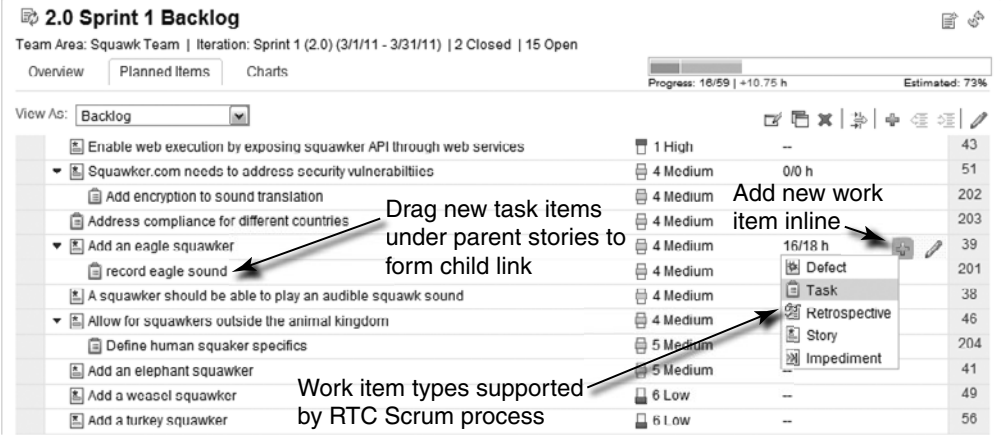


Figure 11.10 Working with the sprint backlog

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.

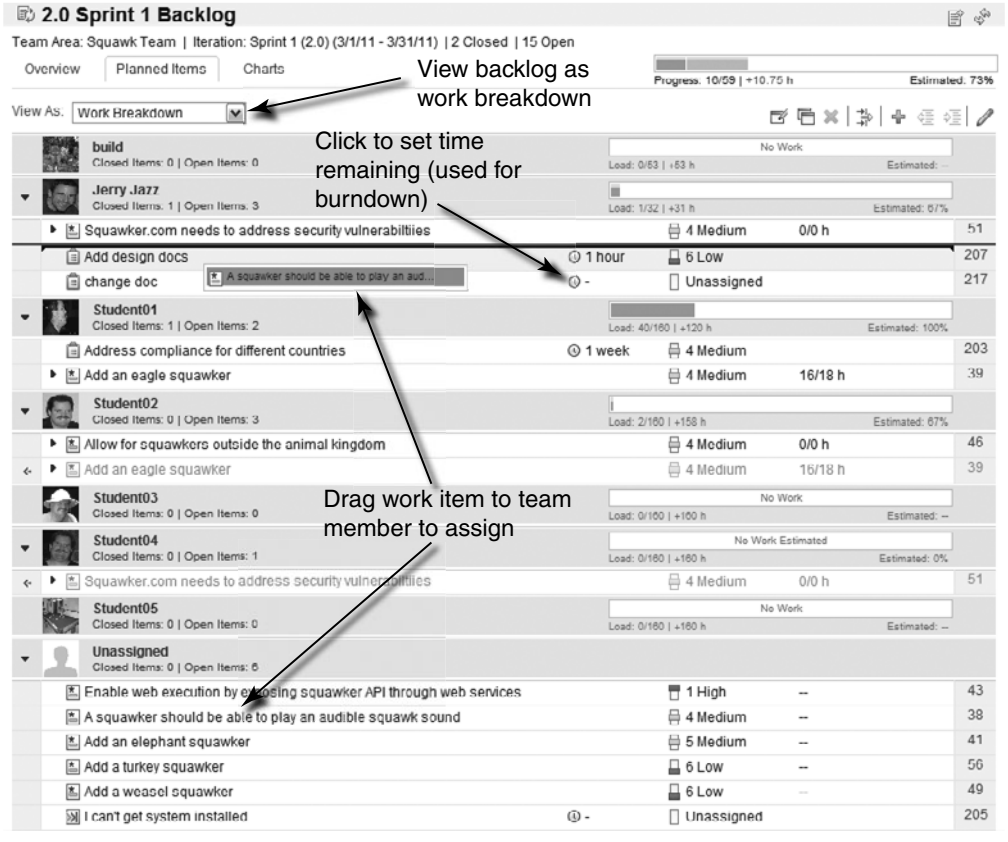


Figure 11.11 Assigning tasks to individuals

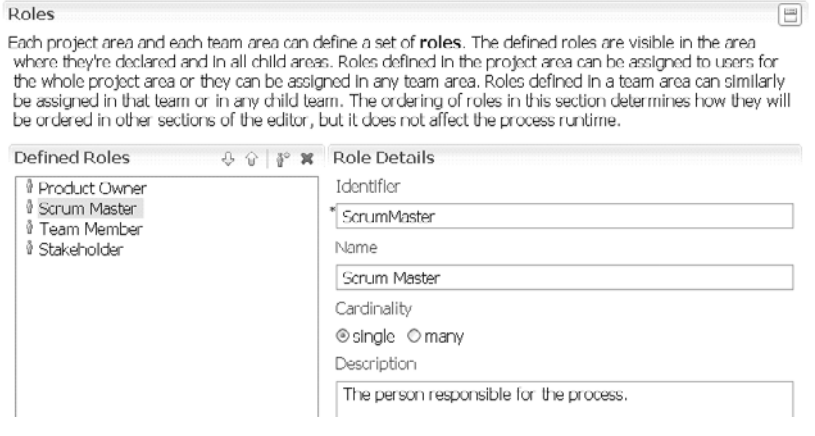


Figure 11.12 Process roles in Scrum

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.

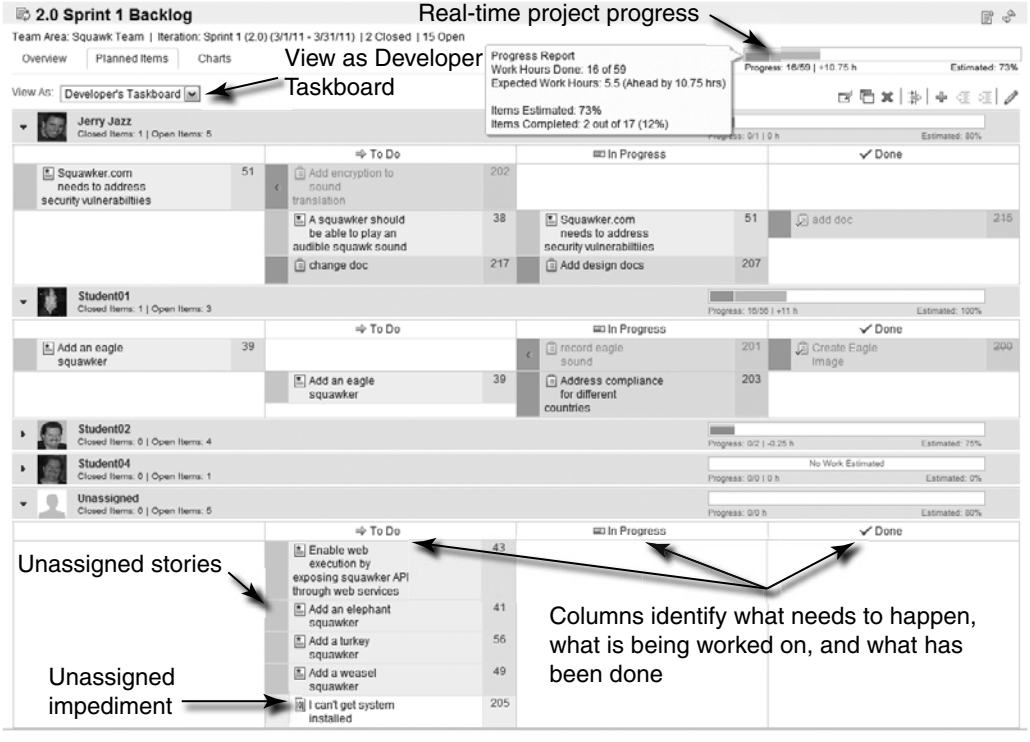


Figure 11.13 Using the Developer Taskboard in the Daily Scrum Meeting

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.

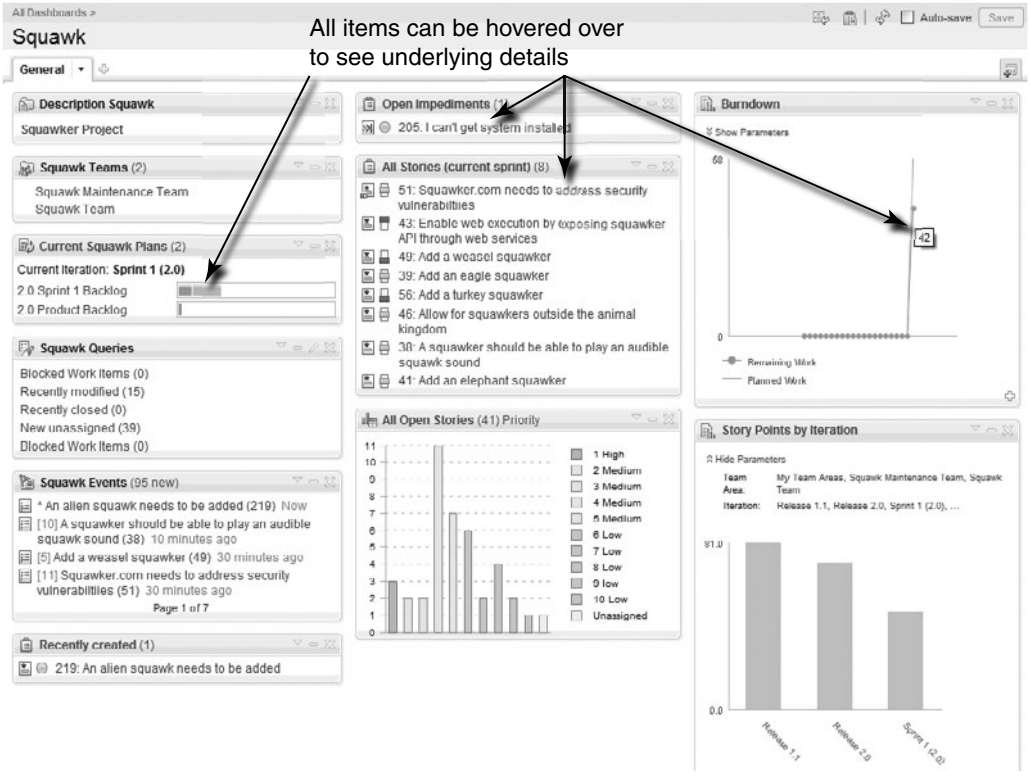


Figure 11.14 Project dashboard showing real-time status of events of interest

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 project 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.

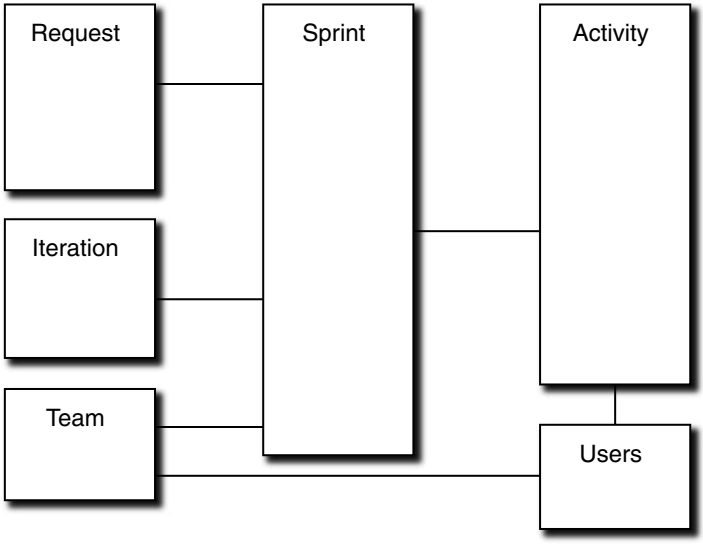


Figure 11.15 Entities relation diagram for the Scrum schema

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.

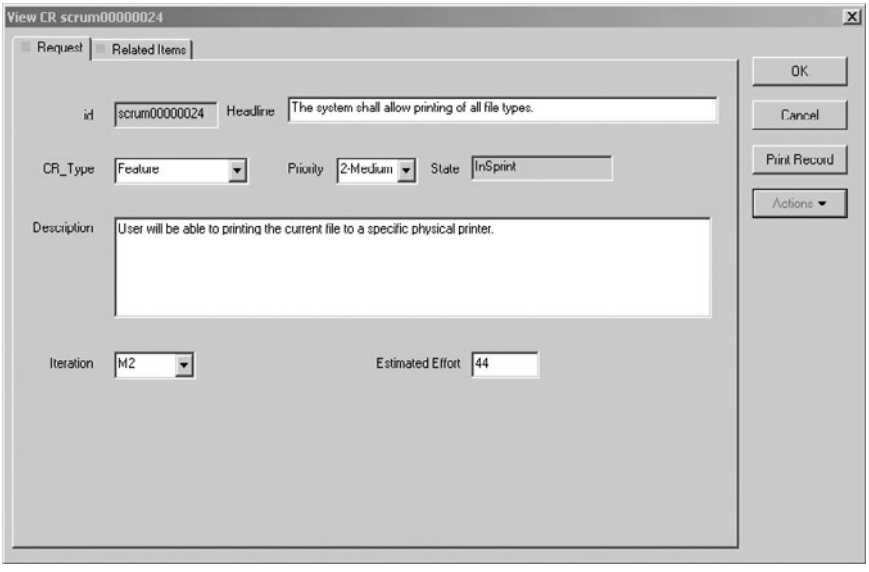


Figure 11.16 Request record: main tab

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.

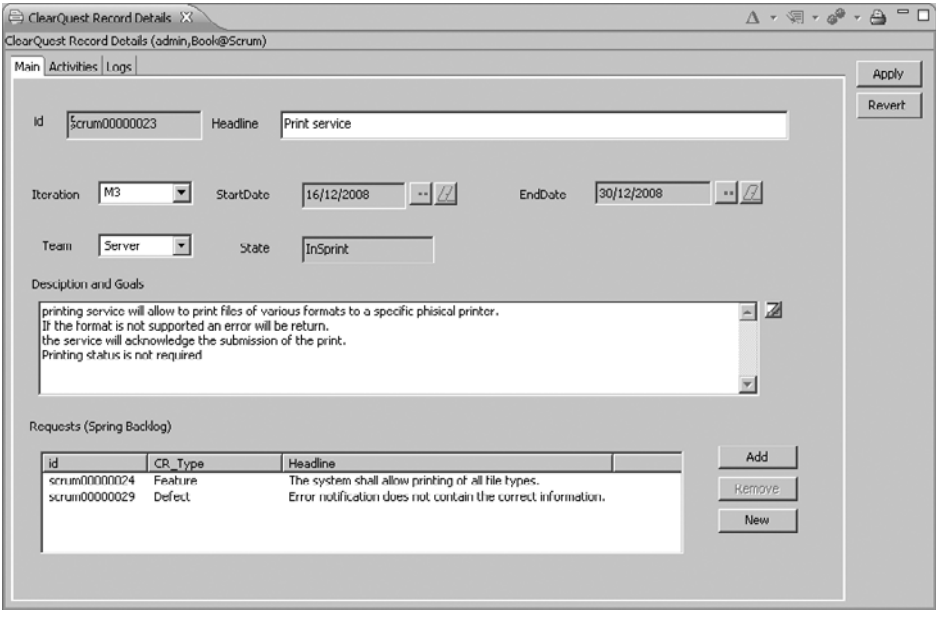


Figure 11.17 Sprint record: main tab

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.

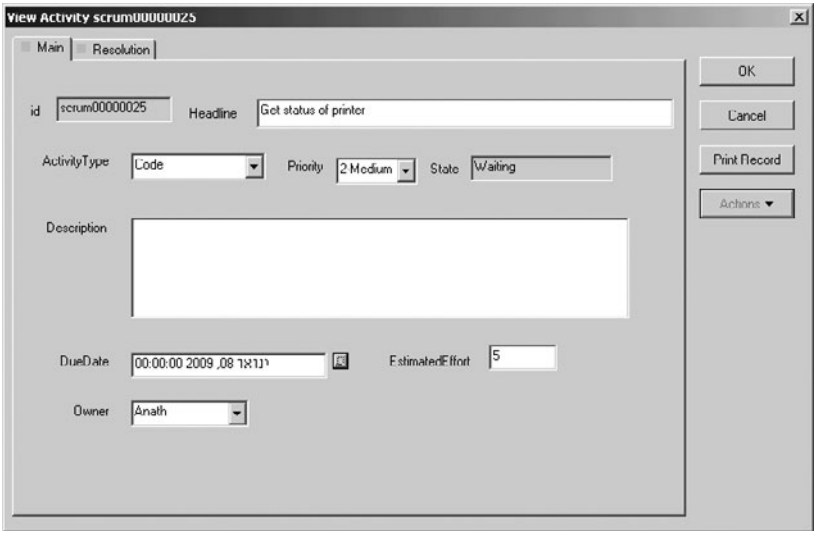


Figure 11.18 Activity record: main tab

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.

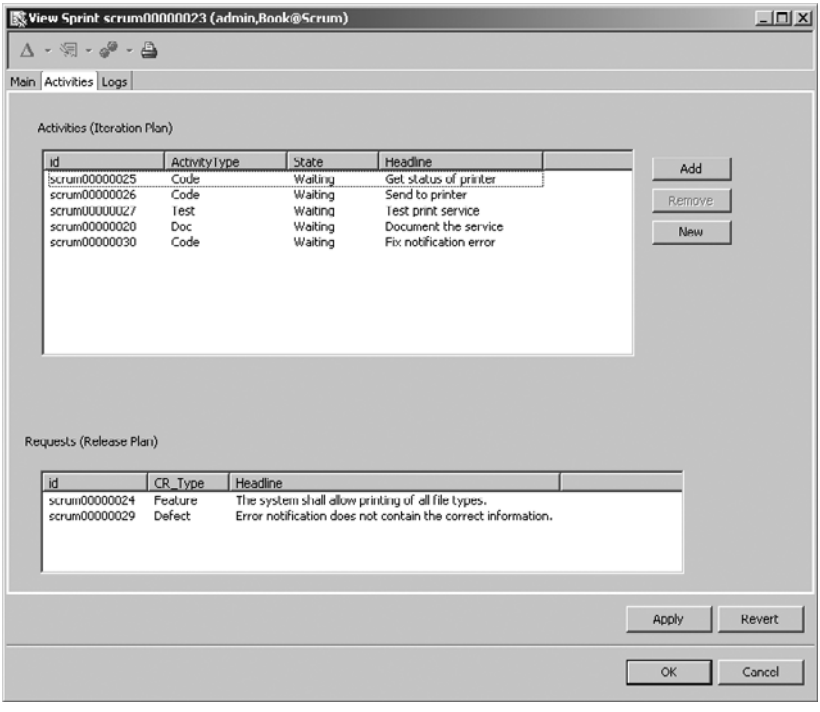


Figure 11.19 Activities and Requests related to a sprint

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.

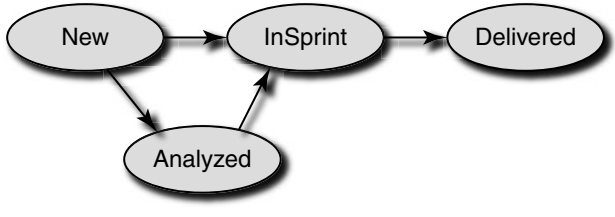


Figure 11.20 The Request record type workflow

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).

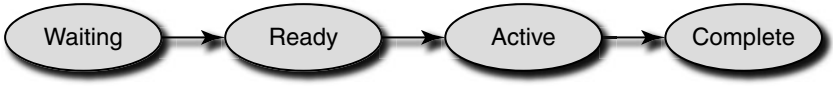


Figure 11.21 The Activity record type workflow

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.

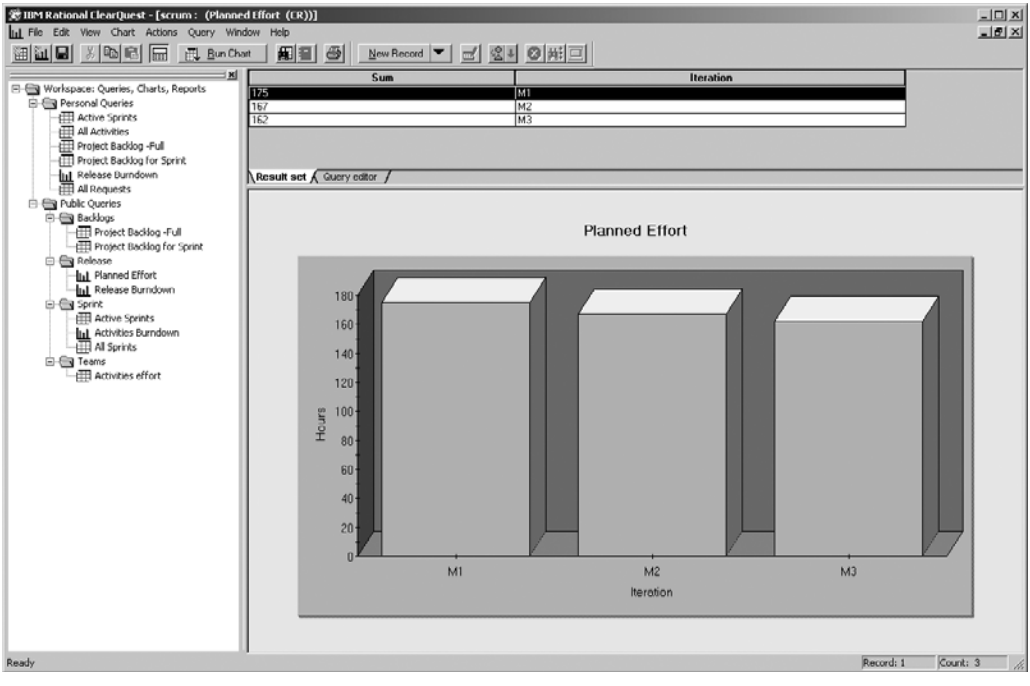


Figure 11.23 Planned Effort for Iteration chart

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.

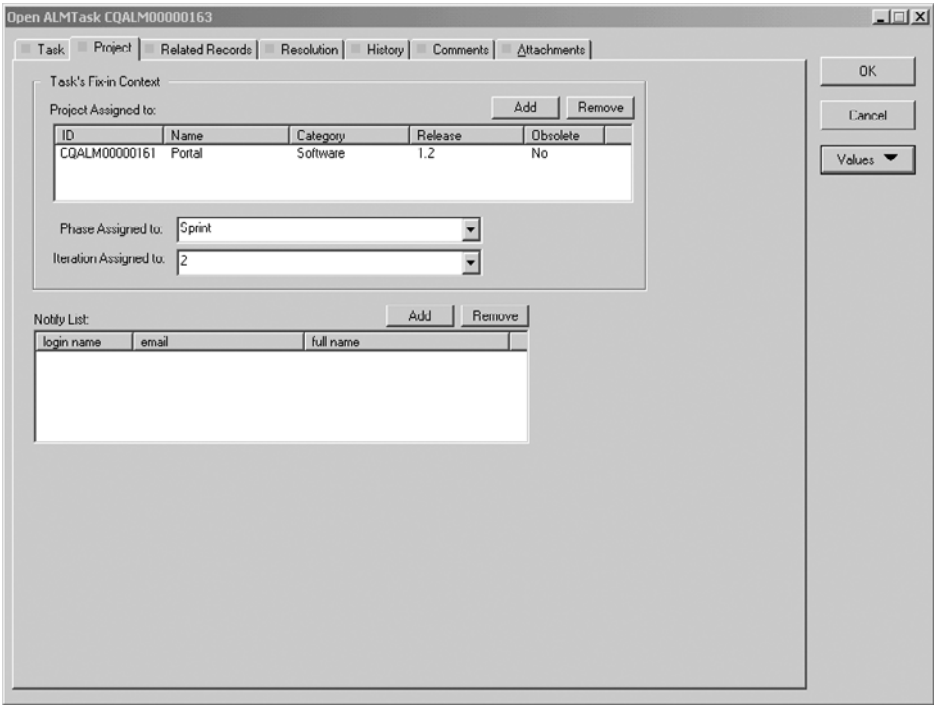


Figure 11.24 ALMTask record assigned to sprint (iteration)

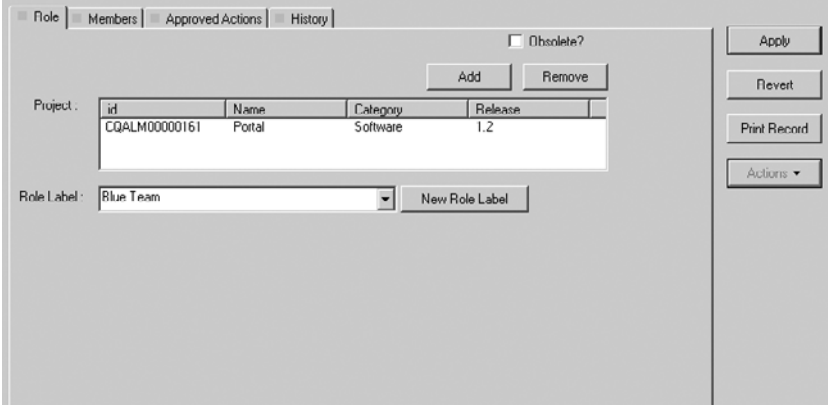


Figure 11.25 ALMRole record assigned to an Agile team

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.

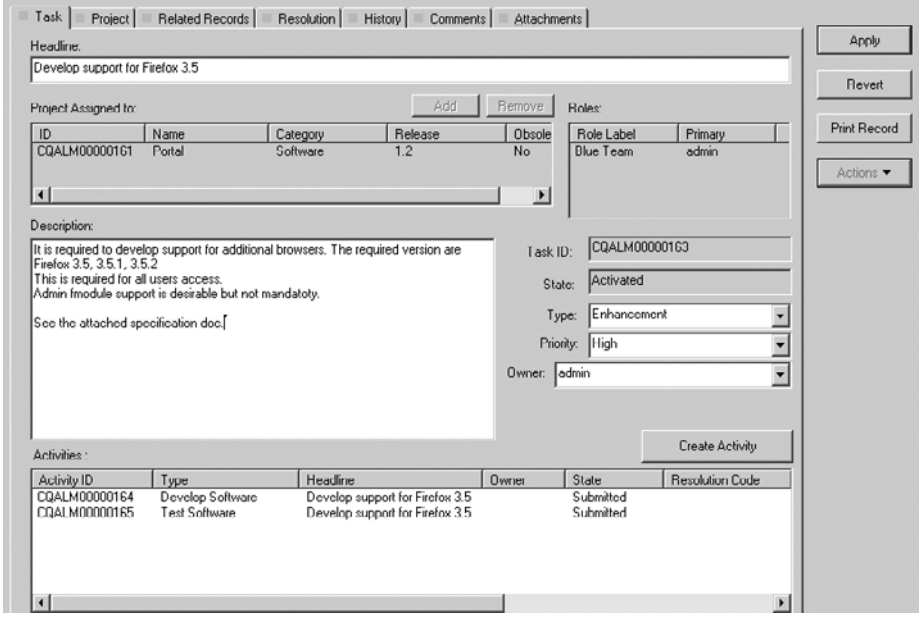


Figure 11.26 ALMTask record assigned to a team

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).

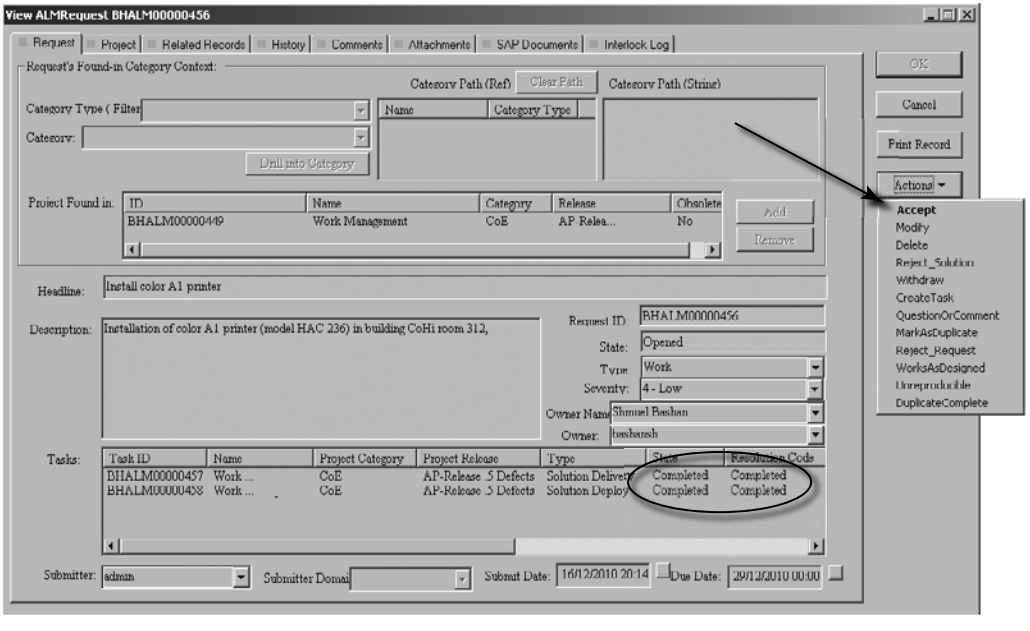


Figure 11.27 ALMRequest solution is accepted by the requestor

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, 2007 (accessed February 23, 2011).

Ambler, Scott, "Scott Ambler's Articles and Other Writings," 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, 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/, 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, 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, 2006 (accessed February 23, 2011).

11.6.3 Other Information

"Agile Alliance," 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/, 2011 (accessed February 23, 2011).

Wells, Don, "Extreme Programming: A Gentle Introduction," www.extremeprogramming.org, 2009 (accessed February 23, 2011).

Wikipedia, "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 developerWorks 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

B

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

X

This page intentionally left blank