CREATING A SOFTWARE ENGINEERING CULTURE

Karl E. Wiegers

CREATING A SOFTWARE ENGINEERING CULTURE

Also Available from DORSET HOUSE

Complete Systems Analysis: The Workbook, the Textbook, the Answers
by James & Suzanne Robertson foreword by Tom DeMarco
ISBN: 0-932633-50-1 Copyright ©1998,1994 624 pages, softcover

Exploring Requirements: Quality Before Design
by Donald C. Gause and Gerald M. Weinberg
ISBN: 0-932633-13-7 Copyright ©1989 320 pages, hardcover

Managing Expectations: Working with People Who Want More, Better, Faster, Sooner, NOW!
by Naomi Karten foreword by Gerald M. Weinberg
ISBN: 0-932633-27-7 Copyright ©1994 240 pages, softcover

***Peopleware: Productive Projects and Teams,* 2nd ed.**
by Tom DeMarco and Timothy Lister
ISBN: 0-932633-43-9 Copyright ©1999 264 pages, softcover

The Psychology of Computer Programming: Silver Anniversary Edition
by Gerald M. Weinberg
ISBN: 0-932633-42-0 Copyright ©1998,1971 360 pages, softcover

Quality Software Management Series by Gerald M. Weinberg

Vol. 1: Systems Thinking
ISBN: 0-932633-22-6 Copyright ©1992 336 pages, hardcover

Vol. 2: First-Order Measurement
ISBN: 0-932633-24-2 Copyright ©1993 360 pages, hardcover

Vol. 3: Congruent Action
ISBN: 0-932633-28-5 Copyright ©1994 328 pages, hardcover

Vol. 4: Anticipating Change
ISBN: 0-932633-32-3 Copyright ©1997 504 pages, hardcover

The Secrets of Consulting:
A Guide to Giving and Getting Advice Successfully
by Gerald M. Weinberg
ISBN: 0-932633-01-3 Copyright ©1988 248 pages, softcover

Surviving the Top Ten Challenges of Software Testing:
A People-Oriented Approach
by William E. Perry and Randall W. Rice
ISBN: 0-932633-38-2 Copyright ©1997 216 pages, softcover

Find Out More about These and Other DH Books:

Contact us to request a Book & Video Catalog and a free issue of *The Dorset House Quarterly,* or to confirm price and shipping information.

DORSET HOUSE PUBLISHING CO., INC.
353 West 12th Street New York, NY 10014 USA
1-800-DH-BOOKS (1-800-342-6657) 212-620-4053 fax: 212-727-1044
dhpubco@aol.com http://www.dorsethouse.com

CREATING A SOFTWARE ENGINEERING CULTURE

Karl E. Wiegers

Dorset House Publishing
353 West 12th Street
New York, New York 10014

Library of Congress Cataloging-in-Publication Data

Wiegers, Karl Eugene, 1953–
Creating a software engineering culture / Karl E. Wiegers.
p. cm.
Includes bibliographical references and index.
ISBN 0-932633-33-1 (hardcover)
1. Software engineering. I. Title.
QA76.758.W52 1996
005.1'068--dc20 96-27627
CIP

Cover Design: Jeff Faville, Faville Design
Cover Photograph: Jim Newmiller

Printed in the United States of America

Library of Congress Catalog Number: 96-27627

ISBN: 0-932633-33-1 12 11 10 9 8 7 6 5 4

To
Pretzels the Clown

This page intentionally left blank

Acknowledgments

The dedicated critiquing efforts of Marcelle Bicker, Tim Brown, Lynda Fleming, Anne Fruci, Kathy Getz, Lesle Hill, Trudy Howles, Kevin Jameson, Tom Lindsay, Lori Schirmer, Doris Sturzenberger, Mike Terrillion, and Nancy Willer led to countless suggestions for improvement. These reviewers caught many errors; any that remain are entirely my responsibility. I also appreciate the helpful comments provided by Linda Butler, Henrietta Foster, Beth Layman, Mark Mitchell, Carol Nichols, Tom Sweeting, and Peter Szmyt.

None of the achievements made by our Kodak software groups would have been possible without the active involvement of all team members. I am indebted to the original members of the Photographic Applications Software Team, who worked with me as we learned how to build better software from January 1990 through April 1993. We learned a lot from each other, built some useful applications, and had fun in the process. Many thanks to Ray Hitt, Tom Lindsay, Mike Malec, and Mike Terrillion. Mike Terrillion also allowed me to describe his work correlating the number of requirements in a software requirements specification with the work effort needed to deliver the application.

Thanks go as well to John Scherer and Jim Terwilliger, managers who didn't know a lot about software, but were willing to learn. They provided solid management support and encouragement during the early years of our process improvement efforts. They also understood the importance of providing recognition for the team's efforts to improve our work and to share our approaches with other software groups at Kodak. Judy Walter and Jeffrey Yu in the Kodak Research Library kept me in touch with the software literature by responding quickly to a steady stream of requests for books and articles.

I am grateful to the staff at Dorset House, particularly Wendy Eakin, David McClintock, Ben Morrison, and freelance editor Teri Monge, for their careful review and editing, insightful recommendations, and support throughout this project.

Special thanks go to Professor Stanley G. Smith of the University of Illinois at Urbana-Champaign, who taught me a lot more in graduate school than how to do physical organic chemistry.

On the home front, I wish to thank my cat, Gremlin, for keeping my lap warm whenever I try to read, and the musical genius of "Weird Al" Yankovic, for helping to put life into proper perspective. My deepest appreciation goes to my wife, Christine Zambito, for her limitless support, encouragement, listening ability, and sense of humor. Living with a clown is just slightly less peculiar than living with a computer person.

Permissions Acknowledgments

The author and publisher gratefully acknowledge the following for their permission to reprint material quoted on the cited pages.

p. 16: Material from *Yeager, An Autobiography* by Chuck Yeager and Leo Janos, p. 235, reprinted by permission of Bantam Doubleday Dell Publishing Group. Copyright © 1985. All rights reserved.

pp. 23, 45: Material from *Peopleware: Productive Projects and Teams* by Tom DeMarco and Timothy Lister, pp. 135 and 12, reprinted by permission of Dorset House Publishing. Copyright © 1987. All rights reserved.

p. 35: Material from *Managing Software Maniacs* by Ken Whitaker, p. 45. Copyright © 1994, published by John Wiley & Sons. All rights reserved.

p. 51: Material reprinted from *Communications of the ACM,* stated purpose—reprinted monthly. Material reprinted from *IEEE Computer,* stated purpose—reprinted monthly.

pp. 61, 214: Material reprinted from *Software Reliability* by Glenford J. Myers, pp. 39, 189–95. Copyright © 1976, published by John Wiley & Sons. All rights reserved.

pp. 61, 165, 313: Material reprinted from *Quality Is Free* by Philip Crosby, pp. 15, 58, 131. Copyright © 1979 McGraw-Hill. Reproduced with permission of The McGraw-Hill Companies.

pp. 63, 147, 158, 168, 170, 215: Material and data adapted from *Assessment and Control of Software Risks* by Capers Jones, pp. 29, 34, 286, 435, 436. Copyright © 1994. Reprinted by permission of Prentice-Hall, Inc., Upper Saddle River, N.J. All rights reserved.

p. 85: Figure 6.3 reprinted from IEEE Std 830-1993 IEEE Recommended Practice for Software Requirements Specifications, Copyright © 1994 by the Institute of Electrical and Electronics Engineers, Inc. The IEEE disclaims any responsibility or liability resulting from the placement and use in this publication. Information is reprinted with the permission of the IEEE.

pp. 105, 127: Material reprinted from W.S. Humphrey, *Managing the Software Process* (pages vii & 287), © 1989 Addison-Wesley Publishing Company, Inc. Reprinted by permission of Addison-Wesley Longman Publishing Company, Inc.

pp. 129, 145, 169: Material and data adapted from CMU/SEI-93-TR-24 and -25 and from the SEI's *The Capability Maturity Model: Guidelines for Improving the Software Process,* Copyright © 1995, Addison-Wesley, by permission of the Software Engineering Institute, Pittsburgh, Penn.

p. 171: Data adapted from *Applied Software Measurement* by Capers Jones, p. 76. Copyright © 1991. Material used with permission of the author and The McGraw-Hill Companies.

p. 181: Material reprinted from *The Art of Software Testing* by Glenford J. Myers, pp. 12–13. Copyright © 1979, published by John Wiley & Sons. All rights reserved.

p. 191: Material reprinted and data adapted from *Software Engineering Economics* by Barry W. Boehm, p. 40. Copyright © 1981. Reprinted by permission of Prentice-Hall, Inc., Upper Saddle River, N.J.

p. 195: Material reprinted from *Software Inspection Process* by Robert G. Ebenau and Susan H. Strauss, p. 39. Copyright ©1994. Material is reproduced with permission of The McGraw-Hill Companies.

p. 200: Data adapted from Kirk Bankes and Fred Sauer, "Ford Systems Inspection Experience," *Proceedings of the 1995 International Information Technology Quality Conference* (Orlando, Fla.: Quality Assurance Institute, 1995). Used courtesy of Kirk Bankes.

pp. 211, 215, 266: Material reprinted and data adapted from *Software Testing Techniques,* 2nd ed. by Boris Beizer, pp. 3 and 75. Copyright © 1990. Reprinted by permission of ITP Computer Press.

p. 244: Material reprinted from James D. Wilson, "Methodology Mania: Which One Fits Best?" *Journal of the Quality Assurance Institute,* Vol. 7, No. 2 (April 1993), p. 22. Reprinted with permission of the Quality Assurance Institute.

p. 263: Figure 15.3 reprinted from James Walsh, "Determining Software Quality," *Computer Language,* Vol. 10, No. 4 (April 1993), p. 65. Reprinted with permission of Miller Freeman Inc.

p. 265: Figure 15.4 adapted from Robert B. Grady, "Practical Results from Measuring Software Quality," *Communications of the ACM,* Vol. 36, No. 11 (November 1993), p. 65. Figure adaptation courtesy of ACM. Copyright © 1993 Association for Computing Machinery, Inc. Reprinted with permission.

p. 316: Ratio provided by private communication from Capers Jones, Chairman, Software Productivity Research, Inc. Used by permission.

p. 326: Figure 20.1 reprinted courtesy of Corporate Ergonomics, Eastman Kodak Company.

Contents

Figures and Tables

Preface

Rarely has a professional field evolved as rapidly as software development. The struggle to stay abreast of new technologies, to deal with accumulated development and maintenance backlogs, and to cope with people issues has become a treadmill race, as software groups work hard just to stay in place. A key goal of disciplined software engineering is to avoid the surprises that can occur when software development goes awry. Software surprises almost always lead to bad news: canceled projects, late delivery, cost overruns, dissatisfied customers, and unemployment because of outsourcing.

The culture of an organization is a critical success factor in its efforts to survive, improve, and flourish. A culture based on a commitment to quality software development and management differentiates a team that practices excellent software engineering from a gaggle of individual programmers doing their best to ship code. In a software engineering culture, the focus on quality is present at all levels—individual, project, and organization.

In this book, I share a cultural framework that was effective in improving the results obtained by several software groups at Eastman Kodak Company. Most of our projects involved small teams of one to five developers, with typical durations of six months to two years. Each part of the book discusses several guiding principles that shaped the way we chose to create software. I also describe the specific software engineering practices that we adopted to improve the quality and productivity of our work. We believe a culture based on these principles and practices has improved our effectiveness as software engineers, the relationship and reputation we have with our customers, and our level of collaborative teamwork. Many of the experiences related and suggestions offered are most relevant to workgroups of two to ten engineers. Since even large software products are often constructed by small teams of engineers working together, these technical activities are applicable in a wide variety of organizations.

With this book I hope to reach first-line software managers, project leaders, and practitioners who wish to drive progress toward an improved, quality-oriented culture in their organization. My goals are to provide practical ideas for immediately improving the way a team performs software engineering, and to show that continuous software process improvement is both possible and worthwhile. I am assuming that the reader has the ability to actually change the culture of his software group, or at least to positively influence those who can drive changes.

I present here a tool kit composed of many ideas and practices for those who wish to improve the quality of the software they develop, along with case studies of how these methods really worked. Our groups have applied all the methods described, and I have used nearly all of them personally. Every anecdote is real, although the names have been changed. While not every team member has used every good method on every project, we invariably obtained better results when we applied these solid engineering practices than when we did not.

An organization grows a quality-directed software culture by blending established approaches from many sources with locally developed solutions to specialized problems. To help point toward useful sources in the voluminous software literature, each chapter provides an annotated bibliography of references and additional reading materials. The references I feel are particularly valuable are marked with a bookshelf icon.

Each chapter contains several "Culture Builder" tips (marked with a handshake icon), which are things a manager or project leader can do to promote an attitude and environment that leads to software engineering excellence. "Culture Killers" are also described, and are marked with a skull and crossbones warning icon. Culture killers are management actions that will undermine a team devoted to superior software engineering or prevent such a culture from developing. Sadly, many of these are real examples. You can probably think of other culture killers from your own experience, as either victim or unknowing perpetrator. Although both builders and killers are written in the form of recommendations, remember that the culture killers are tongue-in-cheek. Don't rush into work next Monday with an agenda of action items selected from the culture killers!

Some of the experiences of our software groups at Kodak were published originally in the following articles; material is included here with permission from the publishers:

Wiegers, Karl E. "Creating a Software Engineering Culture," *Software Development*, Vol. 2, No. 7 (July 1994), pp. 59-66.

———. "Effective Quality Practices in a Small Software Group," *The Software QA Quarterly*, Vol. 1, No. 2 (Spring 1994), pp. 14-26.

———. "Implementing Software Engineering in a Small Software Group," *Computer Language*, Vol. 10, No. 6 (June 1993), pp. 55-64.

———. "Improving Quality Through Software Inspections," *Software Development,* Vol. 3, No. 4 (April 1995), pp. 55-64.

———. "Lessons from Software Work Effort Metrics," *Software Development,* Vol. 2, No. 10 (October 1994), pp. 36-47.

———. "In Search of Excellent Requirements," *Journal of the Quality Assurance Institute,* Vol. 9, No. 1 (January 1995), pp. 23-32.

This page intentionally left blank

Creating a Software Engineering Culture

This page intentionally left blank

Chapter 3

Recognizing Achievements Great and Small

Another major software player decided to "dangle" incentive money in front of its developers. . . . The dictated delivery date came and went and no product had been delivered. . . . Senior management withdrew the incentive and threatened jobs since developers were employed to deliver even without the need for extra compensation. Some of the software engineers (the good ones) left the company, morale was horrible, and by the time a product was delivered, the company's market share had eroded. What a terrible lesson to learn.

—Ken Whitaker, *Managing Software Maniacs*

When I first became the supervisor of the Kodak software group in which I had worked for several years, I initiated a simple (and slightly corny) recognition program. When someone reached a minor project milestone or made a small contribution such as helping another team member with a problem, I gave him a package of M&M® candies, with a message tag attached expressing congratulations or thanks, as appropriate. Bigger achievements generated bigger bags of M&Ms, or something more tangible. It wasn't much, but it was more than we were used to.

As I expected, the candy disappeared immediately, but I was pleasantly surprised to see that some people kept the message tags visible around their desks. To them, the important thing was not the bag of candy, but the words indicating that their manager noticed and valued the progress being made. It soon became apparent that group members preferred to have the presentations made publicly at our weekly team meetings, indicating their desire for peer recognition of even small achievements.

M&Ms worked with our group, but some other social recognition technique might work better for you. We also gave this sort of micro-recognition award to people outside the group who helped us in some way. It brought smiles to their faces and goodwill to our relationships. However you choose to do it, appropriate praise and commendation help to build the culture of teamwork and striving for excellence that we all want, and it can motivate your team members to do an even better job in the future, since they know you appreciate their efforts.

The form and extent of recognition and reward is a visible indication of an organization's culture. If managers believe the employees are lucky just to have jobs, they won't go out of their way to offer even small gestures of appreciation or congratulations. Conversely, in a market characterized by competitive hiring and high staff turnover, an effective recognition program can help retain talented developers. M&Ms won't make up for low salaries or unpleasant working conditions, but simple recognition is an important step in the right direction.

Software engineers are like other people (well, pretty much): We want to be appreciated, and we appreciate being wanted. Besides the internal satisfaction we obtain from interesting, challenging work and the tangible compensation in our paychecks, we want to feel that our efforts are noticed and valued by those around us. We all enjoy receiving compliments, especially when they come from various sources: peers, customers, team leader, senior managers, professional associates.

Praise for a job well done should be timely, direct, personal, and specific (see Fig. 3.1). If you are a manager, don't wait until performance appraisal or salary adjustment time rolls around to pass along some positive feedback. Tell the individual exactly what he or she did well and why you appreciate it. A mumbled "Keep up the good work" in the hallway is more likely to confuse than motivate the recipient. Your team members must know you are sincere when you offer compliments on their work. Though many people feel awkward when they receive a compliment, they appreciate that someone took the trouble to say how pleased he was with your work.

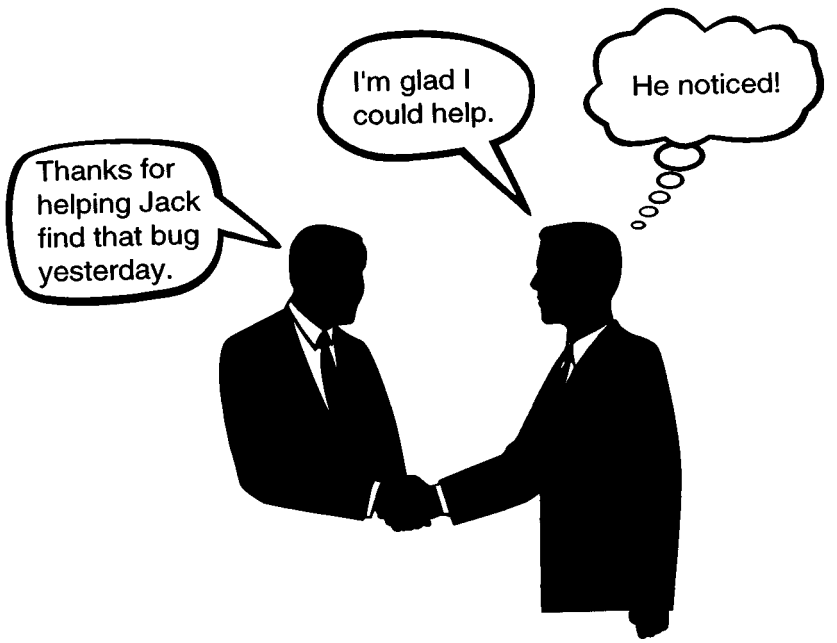

Figure 3.1: The simplest form of recognition.

Recognition can take many forms. Donna Deeprose presents more than one hundred ideas about how to select appropriate and meaningful recognition and rewards for your employees [Deeprose, 1994]. Ask the members of your team what kinds of recognition are important to *them*. Do they prefer a public pronouncement at the weekly group meeting, or are they more comfortable with private ceremonies? Should recognition come just from you, or is it meaningful to have higher-level managers participate in certain recognition activities? Tailor the reinforcement you offer to be significant to the recipients. The following paragraphs describe some of the things the groups in which I have worked have done to express appreciation and to build a positive culture. Some of these apply to individuals, while others are appropriate for teams of people.

Spend a few moments at weekly team meetings to give everyone a chance to pass along some positive reinforcement ("R+") to others. Did a coworker help you solve a problem this week? Did someone take some action out of the ordinary that helped the team? If yes, say so! The group may be uncomfortable when you first try this, but they should warm to the idea over time. If group members are so isolated from each other that no one ever has any R+ to pass along, you may have some serious issues of team dynamics to address.

A traveling trophy that moves from project to project can be used to recognize team achievements. In keeping with the M&M motif, we used a framed three-pound M&M bag (empty, sad to say) as a traveling trophy. Recipients displayed the prize in their office area until another project reached a milestone worthy of recognition. The trophy was ceremoniously passed from one team to the next in our group meeting. If you try something like this, be sure to keep the trophy traveling every few weeks, or its significance becomes lost.

Food and entertainment are also good ways to recognize someone's contributions or special achievement. Taking the team to a celebration luncheon when a milestone is reached can be fun for everyone involved. A gift certificate for dinner at a restaurant gives an individual recipient a chance to celebrate privately with friends or family. Whenever a member of my team earned a college or advanced degree, my wife and I took him and his significant other out to dinner. Maybe going to dinner with the boss is not everyone's idea of a great time, but it worked for us. On another occasion, I gave each team member a pair of movie passes as a symbol of how much I appreciated their time and teamwork when, during an intense period of selecting new members for our group, the entire team of ten pitched in on short notice to participate in the interviews. It was a small gesture but a sincere way of saying, "Thanks for the help, gang."

Recognize individuals outside your group for their contributions as well. It's amazing how much future cooperation you can secure with a simple gesture of appreciation. As the recipient of a few such gestures myself, it always makes me feel good to know that someone really appreciated something I did, however routine it may have seemed to me. We have thanked project customer representatives by taking them to lunch, giving them certificates to hang on the wall, and bestowing restaurant gift certificates upon those who shouldered the most responsibility.

One customer even reciprocated, throwing a lunch bash for the development group. This sort of customer-developer interaction helps build a culture of constructive teamwork.

Be sure to recognize people for attaining minor milestones, as well as when they complete a big project. Interim pats on the back help provide team members with the incentive to keep pushing ahead. Again, it says to the recipient, "Congratulations on making progress toward your goals."

As a manager, you must actively look for recognition opportunities, and seize the moment as soon as you spot one. The manager who realizes an achievement is worthy of formal recognition but waits to figure out what he wants to do about it, and waits to execute his plan, may provide recognition too far removed from the achievement itself to mean much to the recipient. "Oh, so you finally noticed what I did," is a typical unspoken reaction to a belated recognition effort. A manager who expects such opportunities automatically to pop up in front of him will miss many of them, and will not deliver consistent recognition messages. Also remember that managers who fail to reward exceptional contributions are sowing the seeds of discontent. The absence of well-deserved recognition is highly demotivating.

The Importance of Being Visible

The antithesis of being recognized for your achievements is feeling that your managers do not know who you are, what you do on the project, how you do it, or what your contributions are to the company as a whole. When was the last time your supervisor stopped by your office just to say hello and to ask how things are going? How about a visit from a manager farther up the corporate hierarchy?

Some people are uncomfortable with an unannounced manager visit, but others welcome the opportunity to share their concerns and show the boss what they are working on. "Management By Walking Around" is one way managers express interest in the individuals in their organization; it can and should be practiced by managers at all levels. Think about it: If you are a first-line supervisor and you never see your boss in the engineering staff's offices, chances are you'll conclude that he or she is hopelessly out of touch with the group.

People are more motivated to put in extra effort when they know the higher-ups value it. We have all worked for managers who represented the other extreme, having a limited awareness of the group's challenges and contributions. How excited can you get about trying to please such managers? As a supervisor, make sure you really know what the engineers in your department are doing. Who are the contributors, the innovators, the leaders? Who is just along for the ride? The team members will have no confidence that they can get fair performance appraisals if they rarely talk to their supervisor. Your employees need to have adequate opportunities to explain what they do and the problems they face.

When appropriate, a manager should also show interest in professional activities that are unrelated to specific projects. It is discouraging to put in extra time to

prepare a presentation for a local software conference, then not to see your boss's face in the audience. Telling you he is pleased you are doing something extra is not nearly so meaningful as if he actually shows up for the event. Remember all those plays, dance presentations, and concerts you sat through because your children were on stage? Being a good manager demands some of the same actions as being a good parent. It means a lot to know your supervisor cares about what you are doing.

The Importance of Management Attitude

Here's a radical idea: Think of you, as a manager, working *for* the people who report to you, as opposed to the more traditional view of subordinates working for the supervisor. The people you supervise are the customers for your leadership and management services, including

- coaching and mentoring
- setting project goals and priorities
- resolving problems and conflicts
- providing resources
- evaluating performance and providing feedback
- career development
- leading process improvement efforts
- providing technical guidance when appropriate

Give your own people top priority over the demands of others for your time. Your priorities as a manager should be those shown in Fig. 3.2.

Unfortunately, too many managers are busy looking *up* the corporate organization chart, not down. The sequence in Fig. 3.2 is frequently inverted, with the key driver being what you think will make your own boss happy. In a healthy, congruent workplace, the boss should be thrilled if you are meeting the needs of your team and its customers. Not everyone is fortunate enough to work in such an enlightened environment; priorities are usually defined by the perception of who you think you have to please to keep your job.

One way a manager can tell if his priorities are straight is whether he ever receives any recognition from his team members. It meant more to me to get an R+ from one of the people I supervised than to get one from my own supervisor. If you are an individual contributor, remember to thank your managers for special

contributions they make or extra help they provide to you. Managers are people, too, with the same desire to be appreciated that anyone else has.

Figure 3.2: Priorities for the enlightened software manager.

Rewards for a Job Well Done

The skillful manager will use recognition and rewards to reinforce desired behaviors, rather than offering tangible incentives to individuals or teams to achieve specific goals. The main incentive for most software people to go to work each day is the opportunity to work on challenging projects with stimulating colleagues, in an environment that encourages quality work, in which software skills can be applied and extended.

Dangling extra cash as a carrot in front of a software engineer to try to get him to work faster-harder-longer can backfire [Whitaker, 1994]. If the ambitious goals are not met, and developers know the pot of gold at the end of the rainbow won't be forthcoming, how do you keep them motivated? Withdraw the incentives (thereby destroying morale), or renew the incentives (thereby showing that falling short of management's outrageous goals is just as meritorious as achieving them)? Either way, everyone loses. Instead of offering incentives, design a reward program that matches your organization's culture and means something to your team members. Motivate your team through frequent interim recognition activities, and reward them for a job well done when the job really is done. People should also be rewarded for taking intelligent risks, even if a great notion or great effort doesn't

pay off for reasons outside the development team's control. Public rewards indicate to the rest of the group those behaviors you feel are desirable.

Rewards can be monetary or non-monetary. Slipping a few bonus bills in the pay envelope may seem like a sure way to please an employee, but often something else would be preferable. Talk to your people and understand what rewards they feel are significant. The corporate culture will have some influence in selecting feasible rewards. It may be easier to give someone a substantial, but non-cash, award, such as a trip to a conference or trade show. Some companies reward employees with frequent flyer miles, which can be purchased from airlines for a few cents per mile. Some developers might enjoy extra vacation time; others might want to buy a special software package with which to experiment, just because they heard it was interesting. If someone does an exceptional job on a project, he might appreciate a mini-sabbatical, a couple of weeks set aside to work on whatever he likes.

Another kind of reward is the opportunity to work on an exciting new project. In some groups, only the old hands have the skills and knowledge to keep the legacy systems alive. Less experienced people may be assigned to projects involving newer technology, which are more fun than maintaining ancient applications. This is a good way to drive your senior staff members out of the group, to search for opportunities where they can learn contemporary skills and work on the kinds of projects they read about in computer magazines. Creative and experienced engineers don't want to be mired in legacy code for the rest of their careers. Look for ways to reward your best workers by keeping them stimulated with new learning opportunities and project challenges.

Summary

- ✔ People need to feel the work they do is appreciated.
- ✔ Receiving positive reinforcement from peers, managers, and customers is highly motivating to most people. Failing to recognize someone's exceptional contributions and major achievements is demotivating. Why do extra work if your managers don't care?
- ✔ Recognition says, "I appreciate your effort," "Congratulations on your accomplishment," or simply, "I noticed what you did."
- ✔ Find out what kinds of recognition and rewards are meaningful to your people, and tailor your R+ program accordingly to foster a culture of desirable software engineering behaviors.
- ✔ Recognize minor accomplishments and milestones, to motivate individuals to keep working toward their major objectives.

- ✔ Managers can build better relationships with their team members simply by understanding the work they do and showing a sincere interest in it. Talk to your people informally, to find out what they are excited and unhappy about. Sometimes, this approach will let you deal with a concern before it becomes a crisis.

- ✔ A software manager should regard the people who report to him as his most important customers. The manager's top priority should be to address the needs of his team members.

- ✔ Reward your staff members, whatever their job, for a good performance, rather than offering big incentives to induce them to do great work in the future.

Culture Builders and Killers

Culture Builder: Distribute recognition awards equitably to your group members. Don't reserve recognition events only for project leaders, members of high-profile project teams, or your senior technical people. The scale and frequency of rewards does not have to be the same for everyone—after all, people are different—but it is demoralizing for an employee to see the same coworkers being recognized repeatedly without anyone noticing his own achievements.

Culture Builder: Make sure you are accessible to the people who report to you. Schedule one-on-one meetings with those who desire them, at whatever interval is mutually acceptable to you and each individual. A general open-door policy is important, too, but your team members may be reluctant to bother you if they know how busy you are. They deserve a slice of your undivided attention at regular intervals.

Culture Builder: If you make a verbal commitment to someone for a reward, be sure to follow through on it. Forgetting that you made this promise demonstrates a lack of sincerity. Be sure to reward the right people for the right reasons. If you aren't sure who made key contributions to a successful project, find out before you present any rewards or recognition. Few things are more infuriating than seeing a person receive praise (or more) for work that was actually done by someone else.

Culture Killer: In an era of political correctness carried to an extreme, you don't dare run the risk of anyone crying "Discrimination!" on any basis. Therefore, it is safest to offer exactly the same kinds of recognition to

all of your team members, whether they excel in the performance of highly challenging work or they struggle to meet minimal expectations.

Culture Killer: Here are some good reasons to cancel, cut short, or interrupt a regularly scheduled one-on-one meeting with one of your team members:

- Someone else has already stopped by your office to chat.
- You need to work on one of your own projects.
- Your telephone rings.
- You have been invited to join a new committee that meets at that same time.
- Your boss calls another meeting for that time.
- You have to travel to another site to meet with someone else.
- You forgot about it.

Whenever anything short of a true emergency interferes with a scheduled meeting with someone you supervise, you are sending a clear message: "Everything else I have to do is more important to me than you are."

Culture Killer: Offer recognition, such as a luncheon with several important managers, to an individual who has done an ordinary job on an ordinary assignment, but offer nothing to other group members for taking significant initiative that extends outside the boundaries of their assignment. Word will get around that recognition depends on who you are, not what you do.

References and Further Reading

Deeprose, Donna. *How to Recognize and Reward Employees.* New York: AMACOM, 1994.

Deeprose presents ten guidelines that can give your recognition program more impact. She lists one hundred ways you might provide recognition and reward, in the form of structured reward programs, spontaneous rewards, and day-to-day feedback. You can recover the cost of the book the first time you provide an employee with recognition that motivates him to work a little bit harder or smarter on the company's behalf.

Whitaker, Ken. *Managing Software Maniacs.* New York: John Wiley & Sons, 1994.

In Chapter 3, "Attracting and Keeping Developers," Whitaker argues that managers should reward developers after the project is completed, rather than trying to motivate them by promising wonderful rewards in advance as an incentive. This book contains many horror stories of management actions that led to undesirable results.

Author Index

Subject Index