Delivering Successful Projects with TSP℠ and Six Sigma
A Practical Guide to Implementing Team Software Process℠

Mukesh Jain
Delivering Successful Projects with TSP™ and Six Sigma
A Practical Guide to Implementing Team Software Process™
Other Auerbach Publications in
Software Development, Software Engineering,
and Project Management

Optimizing Human Capital with a Strategic Project Office
J. Kent Crawford and Jeannette Cabanis-Brewin
978-0-8493-5410-6

The Business Value of IT
Michael D.S. Harris, David Herron, and Sasia Iwanicki
978-1-4200-6474-2

Best Practices in Business Technology Management
Stephen J. Andriole
978-1-4200-633-2

Effective Software Maintenance and Evolution
Stanislaw Jarzabek
978-0-8493-3592-1

Interpreting the CMMI®, Second Edition
Margaret Kulpa and Kent Johnson
978-1-4200-6502-2

Global Engineering Project Management
M. Kemal Atesmen
978-1-4200-7393-5

Manage Software Testing
Peter Farrell-Vinay
978-0-8493-9383-9

Programming Languages for Business Problem Solving
Shouhong Wang and Hai Wang
978-1-4200-6264-9

Patterns for Performance and Operability
Chris Ford, Ido Gileadi, Sanjiv Purba, and Mike Moerman
978-1-4200-5334-0

The Handbook of Mobile Middleware
Paolo Bellavista and Antonio Corradi
978-0-8493-3833-5

Managing Global Development Risk
James M. Hussey andSteven E. Hall
978-1-4200-5520-7

Implementing Electronic Document and Record Management Systems
Azad Adam
978-0-8493-8059-4

Leading IT Projects: The IT Managers Guide
Jessica Keyes
978-1-4200-7082-8

A Standard for Enterprise Project Management
Michael S. Zambruski
978-1-4200-7245-7

The Art of Software Modeling
Benjamin A. Lieberman
978-1-4200-4462-1

Gerard M. Hill
978-1-4200-4680-9

Building Software: A Practitioner’s Guide
Nikhilesh Krishnamurthy and Amitabh Saran
978-0-8493-7303-9

Software Engineering Foundations
Yingxu Wang
978-0-8493-1931-0

Service Oriented Enterprises
Setrag Knoshafian
978-0-8493-5360-4

Effective Communications for Project Management
Ralph L. Kiem
978-1-4200-6246-5

Software Testing and Continuous Quality Improvement, Third Edition
William E. Lewis
978-1-4200-8073-3

The ROI from Software Quality
Khaled El Emam
978-0-8493-3298-2

Software Sizing, Estimation, and Risk Management
Daniel D. Galorath and Michael W. Evans
978-0-8493-3593-8

Six Sigma Software Development, Second Edition
Christine B. Tayntor
978-1-4200-4462-3

Elements of Compiler Design
Alexander Meduna
978-1-4200-6462-9

Determining Project Requirements
Hans Jonasson
978-1-4200-4502-4

Practical Guide to Project Planning
Ricardo Viana Vargas
978-1-4200-4504-8

Service-Oriented Architecture
James P. Lawler and H. Howell-Barber
978-1-4200-4500-0

Building a Project Work Breakdown Structure
Dennis P. Miller
978-1-4200-6969-3

Building and Maintaining a Data Warehouse
Fon Silvers
978-1-4200-6462-9
Delivering Successful Projects with TSP\textsuperscript{SM} and Six Sigma

A Practical Guide to Implementing Team Software Process\textsuperscript{SM}

Mukesh Jain
3.4 Project Management Myths ............................................................18
3.4.1 Clarifying the Project Management Myths..............................19
3.5 Keys to a Successful Project............................................................21
3.5.1 Goal (SMART Goals) ..............................................................21
3.5.2 Stakeholders (Communicate and Manage) ..............................21
3.5.3 Team (Motivated and Involved) ..............................................21
3.5.4 Project Planning (First Step in the Project) .........................22
3.5.5 Set Expectations (Promise Low and Deliver High) ............22
3.5.6 Design (Know Before You Code) .........................................22
3.5.7 High Quality (Doing It Right the First Time—Everytime) ...23
3.5.8 Iterate and Evolve (Start Small and Build on Top of It) .......23
3.5.9 Testing (Test Early and Frequently) ........................................23
3.5.10 Track (Are We on the Right Path to Achieve the Goal?) ......24
3.5.11 Manage Change (Don’t Let Change Manage You) ..........24
3.5.12 Continuous Improvement (Raise the Bar) .........................25
3.5.13 Positive Attitude (Keep an Open Mind!) .........................25

4 Personal Software ProcessSM (PSPSM) ........................................27
4.1 Overview of PSP ..............................................................................28
4.2 Why PSP .........................................................................................29
4.2.1 Predictability ............................................................................29
4.2.2 Efficiency .................................................................................30
4.2.3 Quality ....................................................................................30
4.3 The PSP Process ............................................................................30
4.3.1 PSP0: The Baseline Process ....................................................32
4.3.2 PSP1: Personal Planning Process ...........................................32
4.3.3 PSP2: Personal Quality Management Process ....................33
4.3.4 PSP3: Cyclic Personal Process ................................................34
4.3.5 TSPSM: The Team Software Process ................................35
4.4 Key Measurements in PSP ............................................................36
4.4.1 Size .......................................................................................36
4.4.2 Time ......................................................................................37
4.4.3 Defects ...................................................................................40
4.5 Software Project Planning and Tracking Using PSP ..............40

5 PSPSM Training ..............................................................................43
5.1 TSP Executive Strategy Seminar (1 day) .....................................43
5.2 Introduction to Personal Process (2 days) .................................43
5.3 PSP for Engineers (10 days) .......................................................44
5.4 Leading a Development Team (Managing TSP Teams; 3 days) ...44
5.5 PSP Instructor Training (5 days) .................................................44
5.6 TSP Coach Training (5 days) .......................................................45
6 Team Software ProcessSM (TSPSM) .......................................................47
   6.1 Overview of the TSP .................................................................48
   6.2 Understanding Team Dynamics.............................................48
   6.3 Building Self-Directed Teams ................................................48
      6.3.1 Team-Building Strategies.............................................48
      6.3.2 Building a Jelled Team ....................................................49

7 TSPSM Launch Process .................................................................51
   7.1 Using the TSP Workbook .......................................................51
   7.2 Project Meetings .................................................................54
   7.3 Roles and Responsibilities of TSP Meetings .........................54

8 Meeting 1
   Establish Product and Business Goals .......................................55
   Meeting Agenda (Discussion Leader) ........................................55
   8.1 Overview .................................................................55
   8.2 Logistics .................................................................56
   8.3 Meeting Discussions—Approach and Positive Attitude ....58
      8.3.1 Discuss with a Positive Attitude ....................................59
      8.3.2 Potential Meeting Discussion Questions .......................59
   8.4 Introduction of Team Members ............................................60
   8.5 TSP and Presentation of Launch Process Overview ...............60
      8.5.1 Purpose of TSP Launch ..............................................60
      8.5.2 TSP Launch Meetings 1–9: An Overview .....................61
      8.5.3 TSP Launch Artifacts ................................................63
   8.6 Marketing Presentation—Product Objectives and Goals ........63
   8.7 Management Presentation—the Business Need and Goals for
      the Product ........................................................................64
   8.8 Goal Priorities .................................................................65
   8.9 Potential Issues and How to Handle Them .........................66
      8.9.1 Management Availability for Opening and Closing a
            TSP Meeting ..............................................................66
      8.9.2 Team Availability Throughout the Launch ....................67
      8.9.3 Team Member and Management Training .......................67
   8.10 Meeting Wrap-Up ............................................................68

9 Meeting 2
   Define Team Goals and Select Team Roles ...............................69
   Meeting Agenda (Discussion Leader) ........................................69
   9.1 Overview .............................................................................70
   9.2 Reviewing Management and Marketing Stated Goals ..........70
   9.3 Establish Team Goals ........................................................71
9.4 Importance of TSP Roles and Responsibilities ....................................... 74
9.5 TSP Roles and Responsibilities ............................................................... 74
  9.5.1 TSP Role: Team Leader ................................................................. 76
  9.5.2 TSP Role: Customer Interface Manager ........................................ 79
  9.5.3 TSP Role: Design Manager ............................................................ 81
  9.5.4 TSP Role: Implementation Manager .............................................. 83
  9.5.5 TSP Role: Planning Manager ......................................................... 85
  9.5.6 TSP Role: Process Manager .......................................................... 87
  9.5.7 TSP Role: Quality Manager ........................................................... 89
  9.5.8 TSP Role: Support Manager .......................................................... 91
  9.5.9 TSP Role: Test Manager ................................................................. 92
  9.5.10 TSP Role: Team Member .............................................................. 95
  9.5.11 TSP Role: Inspection Manager .................................................... 97
9.6 Role Manager Commitments .................................................................. 100
9.7 Selecting TSP Roles ............................................................................. 101
  9.7.1 Role Category ............................................................................. 101
  9.7.2 Guidelines for Selecting TSP Roles ................................................. 102
  9.7.3 Role Selection Process .................................................................. 103
9.8 Entering Role Details on TSP Sheet ..................................................... 105
9.9 Goal Tracking ...................................................................................... 106
9.10 Meeting Wrap-Up .............................................................................. 107

10 Meeting 3
Produce Development Strategy .................................................................. 109
Meeting Agenda (Discussion Leader) ........................................................... 109
  10.1 Overview ......................................................................................... 110
  10.2 Produce Conceptual Design .............................................................. 110
  10.3 Team’s Development Strategy ........................................................... 111
  10.4 List of Products to Produce ............................................................... 112
    10.4.1 The SUMS Worksheet ............................................................... 113
  10.5 Development Process Definition ....................................................... 117
  10.6 Process Plan .................................................................................... 119
  10.7 Support Plan .................................................................................. 120
  10.8 Change Control Board .................................................................... 121
  10.9 Define Role Tasks and Weekly Status .............................................. 121
  10.10 Meeting Wrap-Up ......................................................................... 122

11 Meeting 4
Building Overall Team Plan ....................................................................... 123
Meeting Agenda (Discussion Leader) ........................................................... 123
  11.1 Overview ......................................................................................... 124
    11.1.1 Available Hours in a Week and Task Hours ................................ 124
  11.2 Work Breakdown Structure (WBS) .................................................. 125
11.3 Size Estimation for All Parts of the Final Product ......................... 125
  11.3.1 Team Estimation .............................................................. 125
  11.3.2 Updating SUMS in TSP Workbook .................................... 128
11.4 Determine Overall Project Resources Requirements ...................... 134
11.5 Create List of Tasks ..................................................................... 135
  11.5.1 Development Phase Percentage Time Allocation .................... 136
  11.5.2 Generate Task List ............................................................ 137
  11.5.3 Update Task List with Phase Information ............................ 142
  11.5.4 Review the Updated Task List .......................................... 146
11.6 Weekly Availability of Team Members .......................................... 146
11.7 Generate Overall Team Plan and Review with the Team ............... 151
11.8 Meeting Wrap-Up ....................................................................... 155

12 Meeting 5
  Develop the Quality Plan ................................................................ 157
Meeting Agenda (Discussion Leader) ..................................................... 157
  12.1 Overview .................................................................................. 157
  12.2 Review Quality Goals ............................................................... 158
  12.3 Estimate Defects Injected .......................................................... 159
  12.4 Estimate Phase Yield (Defect Removal Rate) ............................. 162
  12.5 Create and Review the Quality Plan ........................................ 167
  12.6 Meeting Wrap-Up ..................................................................... 173

13 Meeting 6
  Build Individual Plans and Consolidate .......................................... 175
Meeting Agenda (Discussion Leader) ..................................................... 175
  13.1 Overview .................................................................................. 175
  13.2 Allocate Tasks to Team Members .............................................. 176
  13.3 Create Individual Plans ............................................................. 178
  13.4 Load Balance Individual Plans ............................................... 179
  13.5 Incorporate Internal Project Dependencies in the Plan ............ 181
  13.6 Consolidate and Create Team Project Plan ............................ 182
  13.7 Meeting Wrap-Up ..................................................................... 184

14 Meeting 7
  Conducting Risk Assessment .......................................................... 185
Meeting Agenda (Discussion Leader) ..................................................... 185
  14.1 Overview .................................................................................. 186
  14.2 Risk Identification and Assessment ........................................... 186
    14.2.1 Risk Impact .................................................................... 189
    14.2.2 Risk Probability .............................................................. 189
    14.2.3 Risk Priority Assessment ................................................. 189
Contents

14.3 Create Risk Mitigation and Management Plan ...........................................190
14.3.1 Risk Mitigation .................................................................................190
14.3.2 Risk Management .............................................................................190
14.4 Meeting Wrap-Up ...................................................................................193

15 Meeting 8
Prepare TSPSM Project Launch Report for Management Briefing...........195
Meeting Agenda (Discussion Leader) ..............................................................195
15.1 Overview ...............................................................................................196
15.2 Plan the Management Briefing Presentation ..........................................197
15.3 Create Presentation .................................................................................198
15.4 Review Presentation and Prepare for Management Briefing .................199
15.5 Meeting Wrap-Up .................................................................................200

16 Meeting 9
Hold the Management Review ....................................................................201
Meeting Agenda (Discussion Leader) ..............................................................201
16.1 Overview ...............................................................................................202
16.2 Management Briefing of TSP Project Plan .............................................202
16.3 Plan Review, Discussion, Questions, and Changes ..................................202
16.4 Management's Project Plan Approval .....................................................204
16.5 Meeting Wrap-Up .................................................................................204

17 Postmortem and TSPSM Process Review ...............................................205

18 Managing TSPSM Teams ........................................................................207
18.1 Entering Project Data in TSP .................................................................207
18.1.1 Time on Task .....................................................................................208
18.1.2 Defects Data Entry ...........................................................................210
18.1.3 Task Completion ..............................................................................214
18.1.4 Size Information ..............................................................................215
18.2 Project Data Consolidation ....................................................................218
18.3 Weekly Project Status Meetings .............................................................218
18.4 Project Management with Data ...............................................................221
18.4.1 Earned Value ...................................................................................221
18.4.2 Planned versus Actual Hours per Week .........................................222
18.4.3 Defects Metrics ..............................................................................224
18.4.4 Inspection and Review Rates ...........................................................227
18.4.5 Phase and Process Yield .................................................................228
18.4.6 Percentage Time in Phase ...............................................................228
18.4.7 Project Quality Profile ....................................................................230
18.5 Project Reporting ....................................................................................231
18.6 Our Experience .......................................................................................232
### 19 Coaching TSPSM Teams

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.1 The Importance of Coaching</td>
<td>235</td>
</tr>
<tr>
<td>19.2 Individual and Team Coaching and Guiding in Right Direction</td>
<td>235</td>
</tr>
<tr>
<td>19.3 Improving Individual and Team Performance</td>
<td>238</td>
</tr>
<tr>
<td>19.4 Coaching the Team to Understand Their Own Data</td>
<td>239</td>
</tr>
<tr>
<td>19.5 Summary</td>
<td>239</td>
</tr>
</tbody>
</table>

### 20 Overview of Six Sigma

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Six Sigma Approach</td>
<td>241</td>
</tr>
<tr>
<td>Typical Six Sigma Roles</td>
<td>242</td>
</tr>
<tr>
<td>20.1 The Need for Continuous Improvement</td>
<td>242</td>
</tr>
<tr>
<td>20.2 Solving the Right Problem the Right Way</td>
<td>243</td>
</tr>
<tr>
<td>20.2.1 Defects Are Inevitable</td>
<td>243</td>
</tr>
<tr>
<td>20.2.2 Fixing the Root Cause</td>
<td>243</td>
</tr>
<tr>
<td>20.3 The DMAIC Methodology</td>
<td>243</td>
</tr>
<tr>
<td>20.4 Using Six Sigma in Software</td>
<td>244</td>
</tr>
<tr>
<td>20.5 Project Case Study: Improve Product Specification Quality, Completeness and Effectiveness</td>
<td>247</td>
</tr>
<tr>
<td>20.6 Mentoring a Six Sigma Project</td>
<td>248</td>
</tr>
</tbody>
</table>

Index ...........................................................................................................249
Preface

This book gives an overview of Team Software Process℠ (TSP℠) and real-world details about my experience in successfully implementing TSP and Six Sigma in Microsoft. It also shows how geographically widely distributed new teams were able to deliver on time a very high quality product (two thirds of them with zero defects) with a decent work–life balance. This book highlights real-world scenarios that I experienced while coaching development teams on TSP. It also gives some tips and techniques to follow while implementing TSP.

This will be a handy book on understanding how some of the problems faced by the software industry can be solved using TSP. It gives an overview of TSP and shares actual real-world experience in implementing TSP, telling how some mistakes can be avoided to get the best out of a TSP implementation.

In this book, you will see how to effectively manage the development of a software project and deliver it successfully in line with expectations.

When I mention quality to software engineers, they talk about how impossible it is to eliminate defects from software: “Software will have defects no matter what process you follow.” It’s taken for granted, and since nobody acts to improve the situation, we make it horribly true. This goes, too, for software projects; I get the same reaction from project managers and a lot of time it’s the blame game: “Software project got delayed because software engineers did not do a good job in development” or “Software project timeline was unrealistic—it’s the project manager’s fault.” It’s time we review the situation and enable software engineers and project managers to be effective in software development process—end-to-end.

Delivering successful projects means “the ability to produce high quality software within budget and on time—consistently.” Clients expect the software applications to meet their expectations, to be delivered within budget and expected timeline. The project team strives to delight the clients by meeting or exceeding the expectations, attracting new business, and at the same time making a reasonable profit. If you are able to deliver a project that meets both clients’ and project team’s goals, the project can be considered successfully delivered.
Why I Wrote This Book

In my career, I had the opportunity to work in various areas including ISO 9001, CMM/CMMi level 3,4,5, and Six Sigma with software developers, testers, and project managers, finding that the problems are pretty much the same for any organization, groups, teams, or individuals. We needed something to do at the grassroots level to prevent recurring problems. I tried coaching people with some of the techniques of Six Sigma and blending these with software engineering. I started seeing dramatic results. It was the mind set for change and the passion for improvement and excellence that made the difference. This was not possible without the measurement system and framework.

Then I was exposed to TSP/PSP, and I could relate it to what I was doing. It had a good measurement framework and process to achieve a high-quality product and project management. I then started evangelizing it within Microsoft India, and we began seeing good results.

The knowledge I had from implementing TSP/PSP and Six Sigma was with me and a few of my colleagues, then I started doing presentations in several international conferences. I got a good response and several requests for sharing my knowledge. That is when I thought of writing a book to offer these ideas to a wider audience. With the help of John Wyzalek and Auerbach Publications I was able to get this moving, and put my thoughts onto paper in this book.

How Is the Book Organized?

The book is organized in a natural flow starting with current challenges in the software industry, software project management, and then giving an overview of PSP and TSP.

Next, it goes in depth into the TSP launch process and talks about how to go about a launch, with notes on points needing attention—do’s and don’ts, etc. It can be used as a ready reference along with the TSP material provided by SEI.

Within the appropriate chapters, I have also added images and details for using the TSP tool. This can be used as a user guide for working with the TSP tool.

Toward the end of the book the chapter on Six Sigma gives an overview of the methodology and how to blend Six Sigma with the software development processes.

Who Should Read This Book?

This book is intended for professionals who are involved with software development and software process improvement. It will be useful for the following audience: developers, testers/QA, program managers, project managers, TSP coaches, quality assurance engineers, improvement specialists, process champions, etc. With this
book, they will be able to look at their development process from a different angle and will be able to relate TSP concepts to the problems they are facing. This book will serve as a guiding document for them to implement TSP, to avoid mistakes others have made, and to get the best out of TSP. Organizations will be able to save time and money, and at the same time improve the quality of the output and customer satisfaction, driving more business.

There is no prereading required to understand this book. It covers all the basic concepts related to PSP and TSP. This book is not a replacement for formal training related to PSP and TSP, and is not intended to show a radically different approach to software development.
Acknowledgments

I started using Personal Software Process™ (PSP™) and Team Software Process™ (TSP™) in mid 2004, and received good support and guidance from SEI, especially from Watts Humphrey. In 2005, I had the opportunity to host Watts Humphrey in India during his trip there, learning much from his perspective on PSP and how to implement it successfully in large organizations like Microsoft. Several opportunities came up to review our TSP implementation approach with him, and I was able to fine tune it to be more effective. Today, TSP is implemented in a majority of the projects in Microsoft India; it would not have been possible without continued support and guidance from him. Thank you, Mr. Humphrey!

While writing this book, I have referred to SEI’s TSP Launch Material and the SEI’s (James Over) TSP tool.

TSP Launch and TSP Tool are copyrighted material from SEI. For more information please contact SEI directly. http://www.sei.cmu.edu/tsp/get-started.html

PSP and Personal Software Process are service marks of Carnegie Mellon University. TSP and Personal Software Process are service marks of Carnegie Mellon University.

Capability Maturity Model® (CMM®) and Capability Maturity Model Integration® (CMMI®) are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.
About the Author

Mukesh Jain is a principal quality manager at Microsoft, driving quality of service strategy and continuous improvement in online services. This represents over 200,000 servers for over 700 million users worldwide. He has been with Microsoft for more than nine years, implementing Six Sigma, TSPSM/PSPSM, Mistake Proofing (Poka-Yoke), and driving a quality-focused culture. He is a recipient of Microsoft's most prestigious individual excellence award, the “Gold Star,” for four consecutive years. He recently received an honorarium mention from Microsoft’s Chief Strategy Officer, Craig Mundie, for his contribution to building the Quality of Service Program and achieving outstanding results. He is a recipient of the Asia Pacific Leadership award (runner-up), Role-Model, Great People—Leadership award (IGNITE Category), Innovation Award, Solution Excellence, and several quality of service focus awards.

He has authored two “Thinkweek” papers for Bill Gates on performance monitoring and quality management. In 2001, Jain initiated and led Microsoft’s first-of-its-kind Six Sigma project to improve the user’s experience with Outlook performance. Since then he has mentored 14 Six Green Belt Sigma projects, 4 Black Belt projects, and 27 software projects on TSP, achieving a cumulative savings of $3.1 million.

For the past 13 years, Mukesh has worked at key positions as a developer, business analyst, program manager, solutions architect, coach, process and quality manager, and head of quality. He is an experienced coach and has directed and coached programmers and engineering teams to deliver projects on time, with high quality, and within budget. Prior to Microsoft, Mukesh was associated with several multinational corporations, among them Datamatics, Syntel, and Atos Origin, leading project, quality, and program management. His core expertise focuses on managing quality, leading organizations with process maturity, driving predictability (TSP/PSP, CMM, ISO 9000), measurement/metrics programs, guiding
About the Author

continuous improvements (Six Sigma), and shipping world-class products in multinational organizations. In the industry, he is recognized as a leader and coach and has made significant differences in the careers of several people.

Along with a bachelor’s degree in computer engineering and science, he has achieved various certifications that include Certified Standards Professional, TSP Coach, PSP Developer, PSP Engineer, PSP Instructor, CSTE, CSQA, CQM, CQIA, CQA, CTFL, CPD, CPE, Six Sigma Black Belt, Microsoft Office Specialist, ISO 9000 Auditor, MOF, and ITIL. In 2006, he was honored as “Best Six Sigma Black Belt” by iSixSigma magazine.

Mukesh’s work in process improvements, defect prevention, Six Sigma, and TSP/PSP has been recognized at various international conferences, including

- “Quality of Service: Measuring and Improving User Experience the Right Way,” Microsoft (Engineering Excellence/Trustworthy Computing Conference), United States (June 2008)
- “Improving Web Application Performance Using Six Sigma,” BZMedia Software Performance Conference, United States (April 2008)
- “Blending Inspections With Agile and TSP—What’s in it ?” SEI SEPG 2008, USA (March 2008)
- “Improving Web Application Performance Using Six Sigma,” BZMedia Software Performance Conference, Boston (October 2007)
- “Improving Web Application Performance Using Six Sigma,” BZMedia Software Performance Conference, United States (April 2007)
- “Planning for Highly Predictable Results with TSP/PSP, Six Sigma, and Poka-Yoke,” Microsoft Engineering Excellence (November 2006)
- “Planning for Highly Predictable Results with TSP/PSP, Six Sigma, and Poka-Yoke,” PNSQC conference, United States (October 2006)
- “Personal Productivity Improvement with TSP/PSP, Six Sigma, and Poka-Yoke,” NASSCOM, India (September 2006)
- “Improving Product Usability Through Six Sigma,” STeP-In Usability Conference (May 2006)
“Can Your Software Project Deliver High Quality Results Within Budget, On-Time–Every Time?” STeP-Automation Conference, India (June 2006)
“Test Process Maturity thru PSP/TSP,” QAI, India (December 2005)
“TSP in Global Model,” SEI TSP Conference, Pittsburgh (September 2005)
“TSP/PSP,” Microsoft (Trustworthy Computing Conference) (June 2005)
“Six Sigma and TSP,” Microsoft (Corporate Business Excellence Conference) (June 2005)
“Planning for Success with TSP, Six Sigma,” SEI TSP User Group Conference, India (November 2004)
“Reducing Defects by Eliminating Mistakes at Source,” ASQ, United States (March 2002) (Award: Quality Laureate)

Jain has published articles and white papers in several notable magazines including Microsoft, Satyam, ICFAI, and iSixSigma. He has been on the advisory boards for Keynote Systems, PNSQC, IEEE, ISPI, ASQ, SEI, SPIN, and CAI, among others.

Please visit http://www.MukeshJain.org for his latest profile, articles, and white papers.
Chapter 1

Introduction

Today, our global competitive marketplace demands the best of everything. A defect-free product delivered on time at minimal cost is the new standard that demanding customers expect and good suppliers continually strive to meet. In addition to ever more technical challenges, projects are more complex because of geographically distributed teams, time differences, language, and cultural differences. Project members do not have an easy task in meeting these challenges while still delivering high levels of quality, meeting schedules, and managing costs, as well as balancing professional and personal lives. There is no silver bullet, but we can meet all these goals by embracing a disciplined software development process and managing the project with the right set of metrics.

In the current economy, the majority of the products and services we consume interact with or use software. Software is so common in everything that we hardly notice its presence. It is, however, important to remember that a lot of hard work by hundreds to thousands of people goes into the development of these software products.

Often, projects are planned by managers and executed by a team; if things go wrong, it becomes the team’s failure. It is easy to be reactive at work, but much harder and very necessary to be proactive and to plan teamwork. Historically, application development is often estimated and tracked based on a gut feeling. A schedule is developed from the rough estimate and, as the team tries to meet its schedule, it struggles to meet deadlines. Corners get cut, and the last phase—testing and stabilization—usually suffers the full effect of poor planning and estimating. Engineers often have to spend endless evenings and weekends in the office to get the project out, and their work and life become unbalanced, which is not healthy for any organization from any angle, including money, time, and staffing. In the end, customers are most likely dissatisfied, teams are exhausted, and quality may have been compromised in the drive to meet the promised date.
Delivering Successful Projects with TSP\textsuperscript{SM} and Six Sigma

I analyzed several projects and found that more than two-thirds of the effort and money is spent in testing and stabilizing the product. This is a non-value-added activity, and managing it properly can be a great opportunity to help organizations save time and money.

The software engineering’s Team Software Process\textsuperscript{SM} (TSP\textsuperscript{SM}) really shifts the focus from testing as the “find it and fix it” stage. With TSP, each individual engineer acts to prevent defects throughout the project life cycle. Team members record data during project execution, track key metrics, and take corrective action as soon as the project deviates from the plan. Each engineer performs a self review to ensure the quality of their own output before it goes to the next phase. This brings about a high level of predictability in schedule, effort, and quality.

Instead of managers, the TSP team plans the project, tracks all the important metrics on a weekly basis, and takes corrective action as needed. It becomes the team’s project plan, and every member is committed to it. Using Six Sigma along with TSP makes future planning even better. Senior management will be able to strategize more accurately and confidently with a TSP project plan and estimates based on historical data (rather than guesstimates).

This book will detail how TSP techniques can be followed to plan the right thing, do the right thing, and expect the right thing at the right time—every time. I have shared real-world experience in implementing TSP at Microsoft, and now share the success stories on how the newly formed teams had a better work–life balance, delivering very high quality products on schedule and within budget. The projects delivered significant benefits in predictability, and two-thirds of them delivered zero defects. Also, I have highlighted some common pitfalls to avoid while implementing TSP and coaching TSP teams.
Today, the competitive marketplace demands the best of everything—the best quality, reduced costs, and a perfect schedule. A defect-free product delivered on time at minimal cost is the new standard that demanding customers expect and good suppliers continually strive to meet.

Often, projects are planned by managers and executed by the team, and if things go wrong, it becomes the team’s failure. It is easy to be reactive at work but harder and necessary to be proactive and plan your team’s work. Sometimes your manager will grab you in the hall and say, “Hey, can you do this project now and finish it in two months?” Or, a senior management planning committee will call you into its meeting and say, “We need this project now. Can you commit to it?” It is very tempting to say “Yes.” However, saying yes is exactly the wrong thing to do, and your initial “gut feel” estimate will be taken as commitment and the project will be in trouble. Nobody wants to set you and the project up for failure, but, in the rush to get the project done and delight the customer, we all miss the very important step—planning. You could have said, “Let me check to see if my previous estimate is still accurate, and I’ll get back to you within the next two hours.”

It is not an easy task to meet these challenges without compromising the high-quality levels or the schedule or the cost. There is no silver bullet; we can meet all these goals by having a disciplined process, planning it the right way, and managing with the right set of matrices.

In this book, you will learn how sound software engineering principles, project management techniques, and continuous improvement processes can be
Delivering Successful Projects with TSP℠ and Six Sigma

applied to streamline your development process and overcome the crisis that people face due to ad hoc software development processes.

2.1 The Software Crisis

The rapid evolution of the software industry has led people to focus more on project management. Software development, on the other hand, was left to the developers, whose main focus was to deliver the features of the project. Such an approach resulted in an ad hoc approach to software development and no focus on software design concepts or software development methodology. This hampered the application of systematic processes, and can also be referred to as the software crisis.

Some of the major reasons to which you can attribute the software crisis include

- Software developers used multiple programming languages and multiple variations of the languages.
- Most of the requirements were vague and complex and were developed by people who had little or no exposure to software.
- Software developers focused more on technology rather than solution.
- Software engineering education did not give enough emphasis on software quality.
- Software developers poorly mapped requirements to the actual product.
- Getting multiple software systems to work together was a nightmare (very low interoperability).
- Software maintenance was costly.
- Documentation was never given adequate emphasis, which resulted in loss of knowledge and continuity.

2.2 Software Project Management

In the beginning, the focus of software was just to have some program consisting of a set of instructions that would enable the hardware to perform some activity. Most of the commands were mathematical in nature, which the users create themselves for their own use. As the software product grew in importance within organizations, the expectations increased, and more and more people were required to build it. There was a strong need for managing the project and making it more scalable. A project manager was assigned a role to plan the project, manage resources, manage client interaction, and drive software project completion and delivery on time.

The process of software and product development is different from any other project. Traditional methods of measuring size, effort, and productivity cannot be applied to software development. Moreover, techniques to identify, define, and
Measure the software development process and product are comparatively new and
are not mature enough to be used without problems.

Typical software project management activities are

- Understanding and dissipating client requirements
- Determining the scope
- Estimating the cost, effort, expertise and resource requirements
- Identifying and selecting resources with appropriate skills and allocating them
- Identifying the software development life-cycle model
- Identifying risk, creating a mitigation plan, and managing them
- Estimating the project cost and effort required
- Creating project schedule, milestones, deliverables, and overall project plan
- Tracking project schedule and taking necessary action to ensure it would be on track
- Maintaining all versions of the software product
- Managing software product quality, configuration, and changes
- Implementing project tracking, team meetings, communication, and reporting
- Identifying appropriate metrics
- Ensuring project delivery, closure, and post-mortem assessment
- Sharing project learnings with other teams and using them for future projects

Managing a software project is different from managing any other engineering projects due to the complexity and uniqueness of its development process. Although, some of the traditional project management concepts can be applied to software projects, in last 15 years, software project management has evolved as a separate area of expertise.

Typically, a software product is composed of tightly integrated functions, programs, platforms, databases, user interfaces, etc. Several people or teams work together for months and years to develop it. Matters such as cross-group collaboration, attrition, talent-management, people conflicts, managing priorities, and offshore/outsource model add to the complexity, which brings more challenges to project management and chaos in delivering the product successfully. Just having the software that works is not enough; we need to have a quality product that is reliable (that does not have a lot of defects, is resilient to failures, secure, fast, usable, predictable, etc.).

As more and more software applications are being used in financial, medical, and other critical systems, the margin of error is not tolerable, and getting a product to meet this type of quality bar becomes more and more impossible with the traditional approach of software development and project management. You can see, as we move towards more sophisticated systems, expectations are growing, and what worked 5 years back may not work now.
Any projects, similar to any other endeavor, will have one or more important factors regarding successful delivery. Project constraints are scope, schedule, and resources. The iron triangle has these constraints represented as edge (side) of the triangle with quality as a central theme. This is also called Project Management Triangle or Project Management “Triple Constraint,” and is shown in Figure 2.1.

These three constraints are often competing constraints or demands; increased scope typically means increased time and increased cost, a tight time constraint could mean increased costs and reduced scope, and a tight budget could mean increased time and reduced scope. If you change any side of the triangle, one or both of the others will have to change as well. So, if the scope changes, either schedule or resource requirements need to change, or both. One side of the triangle cannot be changed without impacting the others. In this book, you will see the tools and techniques that enable members of the project team (not just the project manager) to organize their work to meet these constraints.

Together, the team can accomplish the project “Right Quality, On Time, On Budget, On Spec.”

### 2.3.1 Scope

The scope constraint refers to what must be done to produce the project’s end result. **Scope** is the set of requirements from the user that is agreed and planned for a particular release (user expectations from a particular system release). It is the overall definition of what the project is supposed to accomplish, and a specific description of what the end result should accomplish. Even though quality may not have been explicitly specified as a requirement, it is a major component of scope of the final product. There are always a set of stated and implied needs from the users. Over the course of a large project, quality can have a significant impact on schedule and cost (or vice versa).
Scope changes are pretty common during the project. In several cases, we need to accommodate the scope changes, as the purpose of the project is the deliver value to the client. Scope changes are of three kinds: new feature, change to a feature, and remove any feature. In any case, the cost is not the same as the original. That is, if a feature requires 100 hours to design and code, changes to it may need more than 100 hours.

### 2.3.2 Schedule

The schedule constraint refers to the amount of time available to complete a project. For all practical purposes, the time required to develop/deliver a project component is estimated using several techniques. One method is to identify tasks/steps needed to produce the deliverables documented in a work breakdown structure (WBS). The work effort for each task is estimated, and those estimates are rolled up into the final deliverable estimate for the project. The tasks are prioritized based on the dependencies between tasks and delivery schedule/milestones. The dependencies between the tasks can affect the length of the overall project, as can the availability of resources. Time or schedule is not considered a cost or a resource because the project manager cannot control the rate at which it is expended. This makes it different from all other resources and cost categories.

### 2.3.3 Resources/Cost

The resources constraint refers to the budgeted amount, machines/equipment, people/consultants, and other resources available for the project. When hiring an independent consultant for a project, cost will typically be determined by the consultant’s or firm’s per diem rate multiplied by an estimated quantity for completion. If you need to finish a job in a shorter time, you can throw more people at the problem, which in turn will raise the cost of the project, unless by doing this task quicker we will reduce costs elsewhere in the project by an equal amount.

### 2.4 Software Quality

The quality of the project end result is not really a constraint but a collective output. The quality of the deliverable is a function of all sides of the triangle. Whenever there is a change in any of the sides, it impacts the overall quality of the deliverable.

There is no point in delivering something very fast and not meeting customer expectations. We cannot take forever either to deliver something that is of exceptional quality. Quality is a subjective measure. If a defect is not encountered, quality is considered high, and yet there may be a defect. So, our goal should be that our users should not encounter any defects. Quality is something that is hard to notice, but its absence/shortcoming is very easily felt.
Software projects rarely follow the planned schedule. Delays happen, and the schedule starts slipping one day at a time. Some of the projects do not have the right level of project tracking, which brings lot of surprises to the development teams towards the end, engineers start cutting corners to meet the date, and the chaos starts. Developers and testing are typically separated; shifting responsibility is illustrated by the attitude that “A test should have caught this problem."

A prevention mindset is not common; instead of analyzing why a bug occurred and finding means to reduce it in the future, the attitude is more often, “The bug is fixed, get a new build.” No effort is expended on finding the root cause of a problem and thinking about preventing this problem in future. By the time the project moves to the testing phase, the project is already in a difficult situation, and managing the project becomes a challenge. The test team is under pressure to complete the test pass and get the project shipped.

Often people have the mindset of testing quality into the project deliverable, the assumption being “Let’s get the coding done, we will put more people to test it and ship it.” Putting more effort in testing does not guarantee better quality. The project should take care of quality right from the planning phase. I will talk more about this in the next few chapters.

I have seen people saying, “Faster, cheaper or better—choose two.” People often think that if you have a tight schedule and a low budget, you will have poor quality, and I have also seen people saying, “Spend more time in testing to get higher quality.” Both are not totally true. It’s like saying, “I can deliver this fast and at a low cost, but it will not meet your needs or may not work.” It is clear that quality should not be part of the constraint, and it can never be an optional component.

If you get your things done right the first time, take all the right measures to ensure quality, you can have an excellent quality product with low budget and a short schedule. On the other extreme, testing can only find certain number of bugs. If you have 100 bugs, and the test process efficiency is 90%, you will still slip 10 bugs. By extending the testing period you might be able to take the test process efficiency to 92–95%, but then you might still have bugs in the end product. The math is clear: to get better quality, focus on the input. If the number of bugs entering the testing phase is 10 (instead of 100), with 90% efficiency you can pretty much ship a world-class product.

People have the misconception that high quality will cost more or will take more time because of variation in its definition. Quality is referred to in multiple ways. It can be broadly classified into three major categories: reliability, usability/aesthetics, and grade. Once the requirements for the product have been agreed upon, its quality refers to the degree to which it meets those requirements.

### 2.4.1 Reliability

Reliability is the ability of a product to perform a required function under stated conditions for a stated period of time. A reliable product will have the attributes of
being accurate, predictable, dependable, secure, maintainable, resilient, and having low or no defects. As you can see, these attributes are important to everybody, and if the product is not able to meet one or more of these attributes and it impacts the customer/user, the product is tagged as a not reliable or poor-quality product.

If the product has defects, unexpected behavior, or raises security or privacy concerns, it will have reliability issues. You can build a reliable product, but not provide good/intuitive UI to access—and your users may not be able to access some of its functionalities at all. So, you can see that all of these aspects are reasonably important, and we need to make sure that they are all there when we talk about quality. Depending on the product and the intended users or customers, we might have different levels of expectations in each of these categories.

In most of the discussion in this book, we will refer to “reliability” when I use the term quality.

2.4.2 Usability/Aesthetics

Just having a good set of functionalities is not enough for a product. The UI needs to be decent and should be intuitive enough to get to this functionality from the main UI. For power users, this may not be very important, but if you are targeting your product to the masses, it assumes considerable significance. Good usability is important for enabling the accessibility features of the product. We want to make sure that the product can be used by everybody, including those who are physically challenged.

If the product usability is not good, people will have difficulty accessing the functionality, and this will increase the support call volume and in turn impact the cost. It might also impact your market share when people find it difficult to use the product. They will tag the product as of poor quality and will seek a better option elsewhere, thus impacting your revenue.

2.4.3 Grade

Grade refers to the set of attributes on which the quality of a product will be judged by some people. For example, you can buy milk in various grades (such as “fat-free,” “1%,” “2%,” “whole,” etc.). The industry has defined a set of attributes for each grade for fat content. The quality of “fat-free” milk will be similar to that of “whole” milk. The grade of the product is based on the scope, and if it meets that need, it will satisfy the appropriate grade quality expectations.

If you want a software product with lot of customization options, then it will cost you more time or money, whereas the same functionality with only a few customization options might be significantly cheaper. An example would be the version of Microsoft Windows Vista or Microsoft Office system; these applications comes in multiple flavors (home, standard, professional, enterprise, ultimate, etc.).

Producing a poor-quality product does not save time or money. In fact, as we will discuss later, quality problems actually cost us time and money.
2.5 Cost of Quality

It is important for us to know and understand the cost of quality and what it covers. Overall, the cost of quality has three components: prevention, detection, and correction.

2.5.1 Prevention

Prevention includes all the activities that can prevent defects from being put in the product in the first place. This includes training, standards and processes, and design activities. For example, requirements engineering, architecture and design activities, design templates, guidelines, automation, coding standards, standard processes, project postmortem, etc., are considered prevention activities.

Most organizations look at these activities as overhead that are not related to delivery of the project, and so they try to minimize them or avoid them altogether. If these activities are correctly included as costs of quality, then they could easily be planned, tracked, and managed to ensure that they are adding value.

2.5.2 Detection

Detection is the set of activities that normally find any defect. Activities such as test planning, test environment setup, test execution, test automation, and reviews and inspection are counted as detection cost. They typically constitute the costs that the organization reports as quality cost. These costs show up in the project budget as the cost of quality.

The usual behavior is, if you want to increase focus on quality, spend more money in the detection area, that is, allocating more resources and time to these activities. I have seen some organizations take pride in saying that they spent more dollars in testing than in development. As you can see, detection activities can only find the defects; you will still need somebody who will have to fix them. You can put more resources in this and find more bugs, but if you do not have adequate budget and time to fix them, you won’t get the benefits of this phase. Also, think about it—if these defects were never there in the product, there would be a huge opportunity to reduce the effort around them.

That is why it is important to consider the other two components of the cost of quality to determine how each of these components interact with each other and how you can drive a positive attitude change in the organization in understanding and appreciating the cost of quality.

2.5.3 Correction

Correction is the set of activities that are triggered by the discovery of defects. It includes defect reporting, and investigating, fixing, retesting, and managing the
defect reports. The major defect correction cost is incurred by development engineers who must investigate and diagnose the problems, devise appropriate fixes, and rework the product to remove the defects. In addition to this, correction includes the cost to test the fix and regression test the system to ensure that the fix did not introduce other problems. Further, if the problem was reported from the field by the customer or user, it includes the cost of distributing the fix and supporting the customers who encounter the problem.

Every time the developer fixes a defect, the overall cost of the project increases and potentially impacts the schedule. However, the increase in project cost and schedule is not tracked as a cost of quality. Instead, retesting is counted as a defect detection cost, hiding it in the wrong part of the cost of quality. To get the correct metrics around this, the process should be in place to mark any rework activity as the correction cost of quality.

I have seen many organizations measure only the cost of detection, and only some organizations measure correction cost. Having a good understanding of the effort in each of the cost-of-quality components is required to make sound decisions for the project and to accurately plan for future projects.

2.6 Global Competition and Market Challenge

With more and more companies adopting the global model, they choose to outsource or set up offshore development offices in countries such as India and China. Some of the reasons for this are cost difference, talent pool, visa restrictions, new and developing market penetration, etc.

New markets bring in a unique set of challenges. Some of them are time difference, cultural difference, language, protocol, and high attrition rate. It becomes more and more challenging to consider these challenges while planning. Just adding more time and people will not help. Acknowledging the differences, and planning for handling these challenges and tracking them is key to success of the project.

2.7 Managing Project Constraints

Different people involved in the project will have different views about the constraints. The QA team considers that quality is the most important attribute to achieve. The marketing team will talk about “feature set,” and hitting a date is a top priority to sustain or capture market share. Management would think that being within budget is very important. So, in the end, everybody is focusing on a different priority, which could result in chaos and impact the shipping of the product. It is very essential to understand what is the right set of priorities for the project and why. The project team needs to be informed about this and empowered to plan the project according to its needs.
A project can be date driven because it needs to
1. Synchronize with another product’s release.
2. Meet competitive threats.
3. Coincide with corporate or consumer buying seasons.

At the beginning of the project, determine if the release date is tied to some external event, and schedule milestone dates accordingly.

A project can be feature-set driven because it needs to
1. Address specific customer requirements and needs that require a minimum set of functionality. This is especially true of version 1.0 products.
2. Address competitive threats.

Whether date-driven or feature-driven, quality is critical and should be considered during the planning stage.

2.8 Project Failures

As said earlier, often projects are planned by managers, executed by the team and, if things go wrong, it becomes the team’s failure. It is easy to be reactive at work. It is harder and necessary to be proactive and plan your team’s work. Sometimes your manager will grab you in the hall and say, “Hey, can you do this project now, and finish it in two months?” Or, a senior management planning committee will call you into its meeting and say, “We need this project now. Can you commit to it?” It is very tempting to say yes. However, saying yes is exactly the wrong thing to do.

Unfortunately, project management is a weak link in the overall software development life cycle. Problems in project management lead to more delays and failures of large software projects than any other known factor. This chapter talks about some of the common causes of project failures and the essential skills for successful software project management and delivery. These skills include budgeting, planning, estimating, measurement, tracking, reporting, and quality assurance and control. Improving the project management process and focusing on sound engineering practices is a critical step towards successful project delivery.

Big projects are usually difficult to control because of the following challenges:

- Hardly anyone has personal plans that provide details of what development will be undertaken, and its measurements and tracking.
- Without such a plan, it is almost impossible for you to find out the status of the work and gain an accurate insight into its schedule.
If we don’t know where you are, we cannot predict or commit to the completion date, and nobody will know the status of the project until it is too late. If management cannot find the current progress of the project and its status, it cannot manage the project.

2.9 Geographically Distributed Teams and Attrition

The majority of the software projects now-a-days spans multiple locations; typically, teams are geographically distributed across the globe. Achieving success with geographically distributed teams is quite a challenge for project managers. As more organizations adopt offshore development, telecommuting, and “going virtual,” this challenge becomes more apparent.

There are benefits and challenges in geo-distributed teams.

Benefits

- More flexibility
- Employee retention
- Wider talent pool
- Cost effective—cost-of-living/salary difference
- Regional knowledge
- Wide range of experience

Challenges

- Communication
- Cultural difference
- Team interaction
- Time difference

Here are some of the characteristics needed for a successful geo-distributed/virtual team.

- A detailed project plan and rigorous project management. All the team members need to have a clear sense of vision, mission and priorities. They know who is responsible for what, and when.
- Ongoing communication. Team members communicate regularly using the phone, conference calls, video teleconference, e-mail, instant messenger, voice/video chat, common databases, SharePoint, Wiki, and other technical methods as appropriate. They must be able to communicate effectively without the benefit of nonverbal cues.
- Free-flowing information. All the team members (new and old, irrespective of the work location) willingly share what they know and what they learn
with everyone else. They also acknowledge or ask about what they don’t know.

- Trust. Team members respect and understand one another as people, not just professionals.
- Cultural factors. Understanding and respecting cultural differences.
- Office visits. The plan should consider regular visits (once every 3–5 months) for team members to be able to spend 1–2 weeks together at one of the locations. This helps in building effective teams, and people can attach faces to names and understand each other better.