Professional Development for Software Organizations

Version 3.01

Construx®

Delivering Software Project Success

11820 Northup Way, Suite E200
Bellevue, WA 98005
(425) 636-0100
www.construx.com

Contents

1. Purpose Of This Handbook

The software industry has developed a number of best practices including software inspections, use of well-defined estimation approaches, controlling changes to requirements, and many other practices. However these practices are not generally well disseminated. As a result, many companies face project overruns, deliver low quality software, burn out their software professionals, or all of the above.

At Construx, we developed a professional development program to address these problems and provide structured career paths to our software employees. As part of our mission to advance the art and science of commercial software engineering, we've created this Handbook to offer an educational resource to Construx clients that will help create a Professional Development Program that is specific to the needs of your organization.

Who Should Use This Handbook

This *Professional Development for Software Organizations* Handbook can be used by anyone who needs to create or enhance a comprehensive professional development plan for their software organization.

Training Managers

Training managers are often assigned the task of researching or creating a professional improvement program for the software staff. While they have expertise in the field of training and development, they lack the subject matter expertise needed to develop a comprehensive program for software employees. If you are in this group, this Handbook will give you recommended topic areas that target high leverage improvement areas.

Software Development Managers

Software Development Managers are experts in software development practices, but often don't know where to start when putting together a professional development program. If you are in this group, this Handbook will help you map out a plan for classroom training, self-study and on-the-job experience.

Individual Software Professionals

Software professionals who are taking charge of their own career development will find this Handbook useful for suggesting reading and career experiences that will give them the necessary skills and knowledge to move ahead.

How To Use This Handbook

We recognize that there is no single best solution for all organizations. Because of this, we recommend that you use this Handbook as a starting point rather than a recipe. This Handbook can be used in its entirety as a template for your own professional development program, or you can choose the components of it that will work best in your organization.

What This Handbook Does And Does Not Do

Software projects operate along four important dimensions: people, process, product and technology. This Handbook focuses on professional development primarily in the software process and people dimensions, i.e., technical and management methodologies. This Handbook does not provide professional development guidelines in the areas of product or technology.

This Handbook contains the following sections:

1. Purpose Of This Handbook
2. Critical Role Of Professional Development
3. Creating A Professional Development Program For Your Organization
4. Whitepaper: Construx Professional Development Ladder
5. Construx Course Catalog
6. How Construx Can Help
7. About Construx

This Handbook is of use to organizations that are creating new professional development programs for software professionals or enhancing existing programs.

Construx's Role In Professional Development

Construx is dedicated to assisting organizations deliver software project success. We achieve this goal by monitoring software industry standards bodies, distilling best practices into real world, useful, practical techniques and processes, dissemination and promotion of software best practices through our consulting, training, software development, professional development and various other services. Our goal in developing this Handbook was to support software organizations that want to improve the way they develop software by improving their staff capabilities.

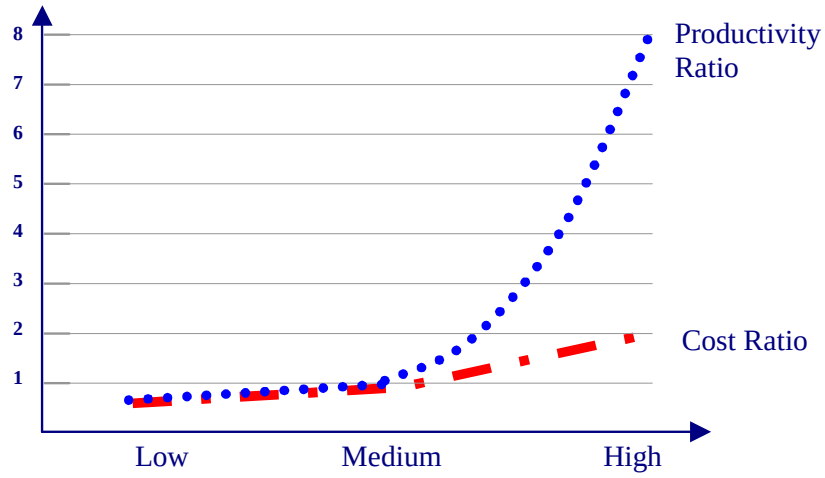
2. Critical Role Of Professional Development

The Case For Professional Development

Industry surveys have found that the average software project is delivered more than 100% late and about 100% over budget.[1] The most common problems that organizations face are poor requirements, inadequate project planning and management, and quality problems that aren't discovered until late in the project.

When an organization commits to professional development, software project performance improves, and there is a direct, positive impact on the bottom line. A study published in *IEEE Software* magazine examined software organization's improvement initiatives in staffing, training, and work environment. The study compared the cost of those programs to the productivity improvements that resulted. The results were that companies that made *Low* or *Medium* investments realized proportionate productivity improvements. Companies that made *High* investments realized a *4-to-1* or *5-to-1 payback.*

The following figure graphically demonstrates those relationships.[2]



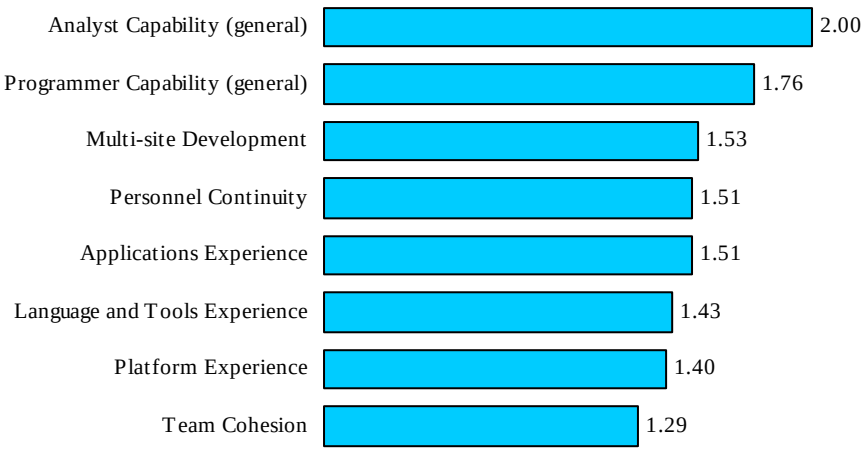
Where's The Leverage?

Research into software estimation has provided valuable insight into levers that managers and organizations can use to improve productivity and quality. Research has consistently found that personnel-related influences exert a tremendous influence on quality and productivity. The graph be-

[1] Standish Group Survey, 2001.

[2] Olsen, Neil C. "Survival of the Fastest: Improving Service Velocity," *IEEE Software*, September 1995, pp. 28-38.

low shows the relative influence on projects of personnel factors from the Cocomo II estimation model. The Cocomo II data shows is that projects staff with requirements analysts who are in the bottom 10% of the profession will cause the *entire project* to require 2.0 times as much effort to develop the same software as would requirements analysts from the top 10% of the profession. Similarly, a project staffed with programmers from the bottom 10% of the talent pool will cause the *entire project* to consume 1.76 times as many resources as a project staffed with programmers from the top end of the talent pool.



Source: Adapted from Software Cost Estimation with Cocomo II, *Barry W. Boehm, et al, Prentice Hall, 2000.*

Technology and Best Practice Training

Two of the factors listed in the graph above relate to specific technologies (platform experience, language and tools experience) but the remaining factors refer to more fundamental skills that span technologies. Construx's focus is on developing fundamental skills that software professionals will carry with them to any and all technologies they work with, both now and in the future.

Why Professional Development?

The average software developer reads less than one professional book per year (not including manuals) and subscribes to no professional magazines.[3] These developers are not developing or advancing themselves professionally. About 75% of these people do not have a degree in computer science or a related field. They learn by trial-and-error and on-the-job training, which means that they risk learning other people's bad habits rather than industry best practices. This method of

[3] DeMarco and Lister, *Peopleware*, 2d Ed, 1999.

professional development perpetuates ineffective, inefficient practices that hinder the success of software projects.

A cohesive Professional Development Program will

- increase your organization's productivity
- reduce development costs
- increase employee retention
- accelerate new employees' learning curve
- improve morale
- help you more accurately budget your training dollars.

Professional Development Self-Assessment

Do you wonder how well your organization supports professional development of its software professionals? The following self assessment can help you answer that question.

Give your organization 3 points for each "true" statement. Give partial credit if you feel there is some degree of accuracy—for example, give 2 points for "mostly true" and 1 point for "somewhat true, but not really." The section following the test explains how to interpret the score. Total the individual scores to get your score. *Note: The term "organization" can refer to your company, department or project team.*

		Score
1.	The majority of our organization's software professionals are able to successfully perform a variety of software development roles, i.e., design and process improvement.	_____
2.	Our organization has processes in place to assist employees in reinforcing and implementing the knowledge and skills they gain through training.	_____
3.	On-the-job training and trial-and-error are not the organization's primary source of professional development.	_____
4.	Our organization supplements formal training with less formal programs such as software engineering discussion groups and mentoring programs.	_____
5.	Our organization can clearly identify which performance problems among our software professionals are due to lack of knowledge or skills.	_____
6.	Our organization knows the major business needs facing our organization and software developers in the next 6 months–2 years, and knows what new skills and knowledge each software developer will need to learn to meet those business needs.	_____

		Score
7.	All software job descriptions in our organization have well-defined knowledge and skills required for the job.	
8.	All software professionals in our organization have clearly defined career path; they know what they have to do to get promoted.	
9.	The majority of our software professionals read at least 4 professional books each year (other than manuals) and regularly read at least two professional journals.	
10.	The majority of our software professionals attend at least 10 days of training per year.	
11.	Our performance reviews reward and encourage formal professional development.	
12.	We have internal training programs to enhance consistent application of software development best practices	
	TOTAL	

Score	Comments
30+ Outstanding	Your organization is highly committed to professional development. Employees know what is expected of them and how to achieve those goals.
25-29 Excellent	An organization at this level is performing much better than average. Software professionals have access to most of the professional development resources they need.
20-24 Good	A score in this range represents a good attempt at professional development. Some employees get the professional development they need to be top performers while others do not.
15-19 Fair	This score is typical. Professional development is hit or miss. It's likely that no one has a complete understanding of the knowledge and skills needed by each individual. An organization with this score experiences problems due to inconsistent skills and knowledge, and high turnover.
<15 Poor	This score indicates that there is no formal professional development in place. A significant amount of resources are wasted due to lack of skills and knowledge. Employees who leave the organization often cite "better career opportunities" as the reason.

The Business Case for Better Software Practices

The material in this section is excerpted from Steve McConnell,* After the Gold Rush: Creating a Profession of Software Engineering, *2d Ed, Addison-Wesley, 2003.

In 1994, James Herbsleb reported that the average "business value" (roughly the same as return on investment) for 13 organizations that had taken on systematic improvement programs was about 5 to 1, with the best organizations realizing returns of 9 to 1.[4] In 1995, Neil C. Olsen reported similar returns for organizations that had made significant investments in staffing, training, and work environments.[5] In 1997, Rini van Solingen reported that the average ROI was 7 to 1 with the best organization realizing returns of 19 to 1.[6] In 2000, Capers Jones reported that the return on investment from process improvement could easily go into double digits (i.e., returns greater than 10 to 1).[7] A recent analysis by Watts Humphrey found that the ROI for improved software practices could be in the neighborhood of 5 to 1 or better.[8]

State of the Practice

Improved software practices are one of the few areas in business in which double digit returns on investment is possible. The reason for this is tied directly to the discussions in Chapters 1 and 2—improved practices have been available for decades, but most organizations aren't using them. Risk of adopting these practices is low; payoff is high. This puts the software industry into an unusual condition.

Most people probably think that software organizational effectiveness is distributed according to a typical bell curve—that is, with a few really poor organizations, a few exceptional organizations, and the majority somewhere in the middle. This is shown in the following figure.

---

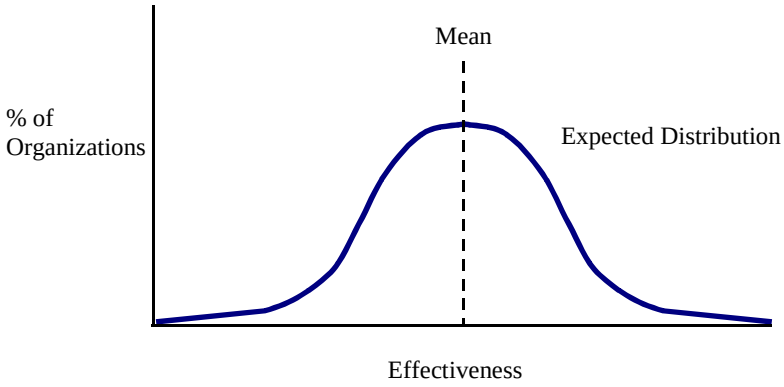
[4] Herbsleb, James, et al, "Benefits of CMM Based Software Process Improvement: Initial Results," Pittsburgh: Software Engineering Institute, Document CMU/SEI-94-TR-13, August 1994.

[5] Olsen, Neil C. "Survival of the Fastest: Improving Service Velocity," *IEEE Software*, September 1995, pp. 28-38.

[6] van Solingen, Rini, "The Cost and Benefits of Software Process Improvement," *Proceedings of the Eight European Conference on Information Technology Evaluation, September 17-18, 2001*.

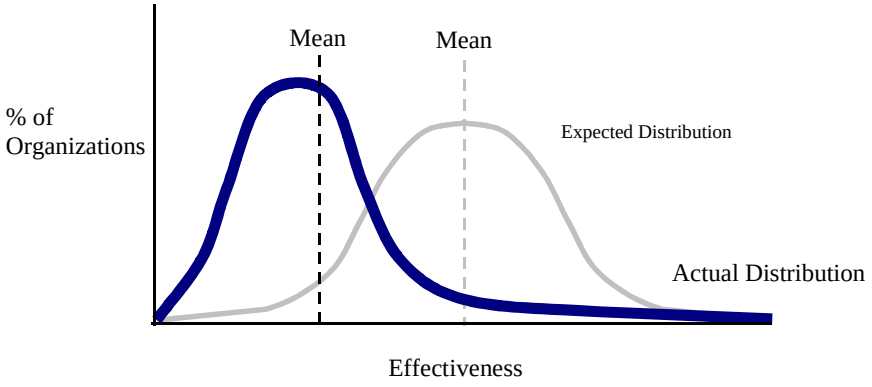
[7] Jones, Capers, 2000. *Software Assessments, Benchmarks, and Best Practices*, Reading, Mass.: Addison Wesley, 2000.

[8] Humphrey, Watts, 2001. *Winning with Software: An Executive Strategy*, Reading, Mass.: Addison Wesley, 2001.



Most people expect organizational effectiveness to be symmetrically distributed, with an equal number of effective and ineffective software organizations.

Contrary to the expectation, the reality is quite different. Due to the slow uptake of effective software practices discussed in Chapters 1 and 2, only a handful of organizations are operating at the highest levels. Industry researchers have long observed a tremendous range between the best organizations operating within an industry and the worst—on the order of 10 to 1.[9] But the average organization is much closer to the least effective organizations than to the best, as shown in the following figure.



The actual distribution of software effectiveness is asymmetric. Most organizations perform much closer to the worst practice than to the best.[10]

The implication of this distribution is that many software personnel have never seen software development at its best. This gives rise to skepticism about whether things are really any better anywhere. As the data in this chapter shows, things are indeed better in the best organizations than they are in the worst.

---

[9] Mills, Harlan D., *Software Productivity*, Boston, Massachusetts: Little, Brown, 1983.

[10] SEI, 2002. "Process Maturity Profile of the Software Community 2001Year End Update," Software Engineering Institute, March 2002.

Detailed Benefits of Improved Software Practices

An in-depth study of 13 organizations by the Software Engineering Institute found that the typical (median) organization engaged in systematic improvement experienced a productivity gain of 35 percent per year, schedule reduction of 19 percent per year, reduction in post-release defect reports of 39 percent per year. These gains provided the underpinnings for the overall returns on investment.[11] These results are summarized in the following table.

Results of Software Process Improvement Efforts[12]

Factor	Median Improvement	Best Sustained Improvement
Productivity	35%/year	58%/year
Schedule	19%/year	23%/year
Post-release defect reports	39%/year	39%/year*
Business value of organizational improvement	5 to 1	8.8 to 1

**The "median" and "sustained" values here are the same because the highest defect-reduction results were short-term improvements and weren't counted as "sustained" improvements.*

The gains experienced by the best organizations were even better. The organization with the strongest productivity gains improved 58 percent per year for four years, for a total compounded gain of more than 500 percent. The best schedule reduction was 23 percent per year for six years, for a total compounded reduction of 91 percent. The best long-term quality improvement was a reduction in post-release defect reports of 39 percent per year for nine years, for a total compounded reduction of 99 percent. Two organizations achieved short-term reductions of 70 percent or more in less than two years.

Organizations that are hooked on code-and-fix development tend to think there is a tradeoff between low defect count and productivity. But as I explained in Chapter 2, much of the cost on a code-and-fix project arises from unplanned defect-correction work. The results in Table 8-1 confirm that for most organizations no tradeoff exists. Organizations that focus on preventing defects also get shorter schedules and higher productivity.

At this point in the industry, dozens of organizations have engaged in systematic improvement in their software practices, and many of those organizations have reported their results in profes-

[11] Herbsleb, James, et al, 1994. *Benefits of CMM Based Software Process Improvement: Initial Results*, Pittsburgh: Software Engineering Institute, Document CMU/SEI-94-TR-13, August 1994.

[12] Data in this table is from Herbsleb, James, et al, 1994. *Benefits of CMM Based Software Process Improvement: Initial Results*, Pittsburgh: Software Engineering Institute, Document CMU/SEI-94-TR-13, August 1994.

sional journals, conference proceedings, and other publications. The following table summarizes the returns on investment reported by more than a dozen organizations.

Examples of Software Improvement ROI[13]

Organization	Results
Boeing Information Systems	Estimates within 20%, $5.5 million saved in 1 year, ROI 7.75
BDN International	ROI 3.0
Computer Sciences Corporation	65% reduction in error rates
Harris ISD DPL	90% defect rate reduction; 2.5x productivity gain
Hewlett-Packard SESD	ROI 9.0
Hughes	$2 million annual reduction in cost overruns, ROI 5.0
IBM Toronto	90% reduction in delivered defects, 80% reduction in rework
Motorola GED	2-3X productivity improvement, 2-7X cycle time reduction, ROI 6.77
Philips	ROI 7.5
Raytheon	ROI 7.7
Schlumberger	4X reduction in beta test bugs
Siemens	90% reduction in released defects
Telcordia	Defects 1/10 industry average, customer satisfaction increased from 60-91% over 4 years
Texas Instruments – Systems Group	90% reduction in delivered defects
Thomson CSF	ROI 3.6
U.S. Navy	ROI 4.1
USAF Ogden Air Logistics Center	ROI 19

[13] Adapted from Krasner, Herb, "Accumulating the Body of Evidence for the Payoff of Software Process Improvement – 1997," November 19, 1997. Unpublished paper; and van Solingen, Rini, "The Cost and Benefits of Software Process Improvement," *Proceedings of the Eight European Conference on Information Technology Evaluation, September 17-18, 2001*.

Organization	Results
USAF Oklahoma City Air Logistics Center	ROI 6.35
USAF Tinker Air Force Base	ROI 6.0

ROIs for Selected Practices

Details of how organizations have achieved their greater software development effectiveness vary among different organizations. Nonetheless, some specific practices have been found to be generally effective. The following table shows the ballpark ROIs for a few selected practices.

Return on Investment for Selected Software Practices[14]

Practice	12-month ROI	36-month ROI
Formal code inspections	2.5	12
Formal design inspections	3.5	10
Cost and quality estimation tools	2.5	12
Long-range technology planning	1.0	10
Productivity measurements	1.5	6.0
Process assessments	1.5	6.0
Management training	1.2	5.5
Technical staff training	0.9	5.0

As with automobile performance, your actual mileage with these practices will vary.

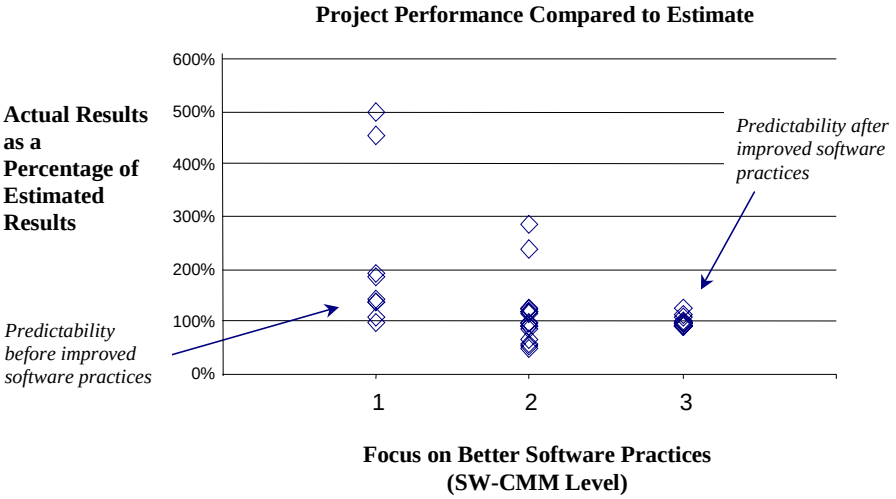
Insights from Software Estimation

One of the indicators of how much control an organization has over its projects is the accuracy of its estimates. A Standish Group survey of more than 8,000 business systems projects found that

[14] Jones, Capers, 1994. *Assessment and Control of Software Risks*. Englewood Cliffs, N.J.: Yourdon Press, 1994.

the average project overran its planned budget by more than 100 percent.[15] The level of estimation error reported in this survey is consistent with other industry findings.[16]

The figure below shows one finding of a study of U.S. Air Force projects at different levels of software practices. (This analysis is based on the SW-CMM, which I'll discuss more in Chapter 10.) Each point below 100 percent represents a project that finished under budget. Each point above 100 percent represents a project that overran its budget.



As organizations improve their software practices, they gain more control over their project estimates, which is generally indicative of more project control.[17]

As you can see from the figure, the least sophisticated projects (SW-CMM Level 1) routinely overran their projects' budgets—in other words, they routinely underestimate their projects' costs. More sophisticated organizations (at SW-CMM Level 2) spread their estimation error more evenly between overestimation and underestimation, but the estimation error was still commonly 100 percent or more. For the most sophisticated organizations (those at SW-CMM Level 3), overruns and under runs become equally common, and the estimation accuracy was much improved.

Indirect Benefits of Improved Software Practices

The return on investment figures in the published literature have mostly been based on operational savings, that is, on reducing development cost per line of code written or per function point deliv-

[15] Standish Group, The, 1994. "Charting the Seas of Information Technology," Dennis, MA: The Standish Group, 1994.

[16] See, for example, Jones, Capers, 1996. *Patterns of Software Systems Failure and Success*. Boston: International Thomson Computer Press, 1996.

[17] Adapted from Lawlis, Dr. Patricia K., Capt. Robert M. Flowe, and Capt. James B. Thordahl. "A Correlational Study of the CMM and Software Development Performance," *Crosstalk*, September 1995.

ered. While these savings are impressive, the greater business benefit might arise from the significant *indirect* returns that arise from improved software practices. Improved software practices improve cost and schedule predictability, reduce risk of cost and schedule overruns, provide early warning of problems, and support better management.

Many organizations that have focused on improving their software practices have reported improvements in predictability similar to the results shown in the figure above. For a software products company, what is the business value of improving schedule estimation accuracy from ±100% to ±10%? What is the value of being able to make a commitment to customers 6–12 months in advance of a scheduled completion date with high confidence of delivering on that commitment?

For a company that develops custom software, what is the business value of being able to provide a fixed price bid with high confidence that the project will not significantly overrun the bid?

For a retail sales organizations, what is the value of being able to plan cutover to a new system with pinpoint accuracy? What is the value of knowing with confidence that cutover will occur October 1, as planned, with little risk of overrunning into the holiday sales season?

Unlike the operational benefits that most of the industry literature has focused on, the indirect benefits of improved software practices open the door to additional revenue opportunities. For top decision makers in organizations, these indirect benefits are often more compelling than the direct, operational benefits.

Challenge is Organizational

Many organizations push responsibility for software development improvement down to the project level. In reviewing the "effort multiplier" factors in the Cocomo II estimation model[18] recently, I was struck by how few of the factors are under the control of an individual project manager. Of the 22 factors Cocomo II uses to fine tune a project's base effort estimate, only 3 in my judgment are typically under the control of an individual project manager (the factors of level of documentation, architecture and risk resolution, and development for reuse). Numerous factors are dictated by the nature of the company's business (product complexity, required reliability, platform volatility, unprecedented ness of the software, and so on). These factors cannot easily be changed without changing businesses. The remaining factors can be influenced by the organization, but cannot readily be influenced by individual projects. These factors include staff capability, multi-site development, personnel continuity, process maturity, and other factors. These organizational factors seem to be where the greatest leverage lies for improved software practices.

[18] Boehm, Barry, et al, 2000. *Software Cost Estimation with Cocomo II*, Reading, Mass.: Addison Wesley, 2000.

3. Construx Professional Development Ladder

A White Paper Available at:

www.construx.com/docs/member/CxLadderWhitepaper.pdf

4. Creating A Professional Development Program For Your Organization

Establishing a professional development program for your organization demonstrates that your employees are valued and that the organization realizes the payoffs for learning how to develop software better. This section outlines the basic activities needed to create a professional development program, lists resources available to support such a program, and provides sample development plans for three common job titles.

Basic Activities

A cohesive professional development program includes the following activities:

1. Identify business and performance goals, both short and long-term.
2. Establish a consensus that professional development has real benefits worth the investment.
3. Create job descriptions that specify the knowledge, skills and abilities for each position based on business and performance goals.
4. Conduct a training needs assessment.
5. Identify software development best practices that will help you achieve organizational goals.
6. Identify training resources that are available to your organization; include books, articles, classes, mentors, and other resources.
7. Develop organization-wide and individual professional development plans.

Each of these steps involves significant supporting detail.

Construx consultants are available to support organizations as they navigate through the details of these steps.

Resources

- Software Engineering Institute (SEI). The SEI provides numerous resources for software development organizations at its website at www.sei.cmu.edu.
- Software Engineering Body of Knowledge (SWEBOK) Project. The SWEBOK provides a cohesive definition of software's body of knowledge at www.swebok.org.
- Project Management Institute (PMI). The PMI provides numerous resources for project managers generally (rather than software specifically) on its website at www.pmi.org.
- Construx's web site provides lists of recommended reading for several software professional job descriptions at www.construx.com.

Sample Professional Development Programs

The first step in creating a professional development program is identifying the *knowledge and skills* that an individual needs to be successful on the job. The second step is identifying the *experience*, *training*, and *reading* that will help the employee gain the skills and knowledge. The third step is to develop personalized professional development programs for each software professional that will raise their skill to the level needed to perform effectively.

The following pages provide additional information and examples of professional development paths for three software project roles: Project Manager, Developer and Tester. These are included to provide an indication of the level of detail that supports effective professional development.

Construx has developed detailed development paths for numerous other roles, and has also developed plans that support multi-stage skills development at the apprentice, practitioner, and leadership levels. These materials are available from Construx in conjunction with our consulting and training offerings.

Project Manager

Knowledge and Skills

In Construx's judgment, project managers need the following set of knowledge and skills to perform their roles effectively:

- ❑ Multiple Estimation Techniques
- ❑ Multiple Development Lifecycles
- ❑ Risk Management Techniques
- ❑ Resource Balancing
- ❑ Basic Configuration Management
- ❑ Basic Change Control
- ❑ Tracking Mechanisms
- ❑ Staffing
- ❑ Project Planning
- ❑ Decision Making
- ❑ Written and Verbal Communication
- ❑ People Management

Professional Development Plan

The following sections describe the experience, training, and reading that will support development of the knowledge and skills necessary to perform effectively as a project manager.

Experience

- ❑ Implement change control on one or more projects

- ❑ Review the configuration management plans for one or more projects
- ❑ Act as a reviewer to at least one project plan
- ❑ Write two or more project plans
- ❑ Create a project estimate
- ❑ Perform tracking for at least one project
- ❑ Perform personal planning and tracking
- ❑ Act as reviewer for a Requirements Specification
- ❑ Act as backup Requirements Lead for at least one project
- ❑ Act as backup Project Manager or Planning and Tracking Lead for at least one project
- ❑ Act as Project Manager on at least one medium sized project

Training

- ❑ Rapid Development
- ❑ Software Estimation
- ❑ Software Project Survival
- ❑ Effective Software Project Management
- ❑ Risk Management
- ❑ Real World Requirements
- ❑ Inspections
- ❑ Peer Reviews for Higher Quality and Productivity
- ❑ People Skills*
- ❑ Negotiation Skills*

*Courses not taught by Construx Software

Reading

- ❑ "Software Engineering Code of Ethics and Professionalism," ACM/IEEE-CS
- ❑ *Software Project Survival Guide*, Steve McConnell
- ❑ *Mastering the Requirements Process*, Robertson and Robertson
- ❑ *Mythical Man-Month*, Fred Brooks
- ❑ *Peopleware*, DeMarco and Lister
- ❑ *Rapid Development*, Steve McConnell
- ❑ *201 Principles of Software Development*, Alan Davis
- ❑ *Manager's Handbook for Software Development*, NASA Goddard Space Flight Center
- ❑ *Project Engineering Body of Knowledge*, PMI
- ❑ *Measures for Excellence*, Lawrence H. Putnam and Ware Myers
- ❑ *Principles of Software Engineering Management*, Tom Gilb
- ❑ *Software Runaways*, Robert Glass
- ❑ *Practical Software Metrics for Project Management*, Grady Booch
- ❑ *The Deadline*, Tom DeMarco
- ❑ "Software's Chronic Crisis," Wayt Gibbs

Developer

Knowledge and Skills

In Construx's judgment, software developers need the following set of knowledge and skills to perform their roles effectively:

- Estimation Techniques
- Risk Mitigation Techniques
- Configuration Management
- Refactoring
- Construction Methods
- Peer Review Methods
- Thorough Algorithm Understanding
- Standard Construction Practice
- Design Decomposition Methods
- Memory Usage Methods
- Modeling
- Written and Verbal Communication
- Interpersonal Skills

Professional Development Plan

The following sections describe the experience, training, and reading that will support development of the knowledge and skills necessary to perform effectively as a software developer.

Experience

- Act as a reviewer on one project's code reviews
- Act as developer on at least one project
- Act as design or implementation lead for a project
- Create at least one accepted design
- Act as a reviewer for a design
- Develop one analysis model
- Review a requirements document

Training

- Object Oriented Analysis and Design using the UML
- Real World Requirements
- Real World Software Testing
- Steve McConnell's Code Complete
- Peer Reviews

Reading

- "Software Engineering Code of Ethics and Professionalism," ACM/IEEE-CS

- *Software Project Survival Guide*, Steve McConnell
- *Mastering the Requirements Process*, Robertson and Robertson
- *Mythical Man-Month*, Fred Brooks
- *Peopleware*, DeMarco and Lister
- *Rapid Development*, Steve McConnell
- *201 Principles of Software Development*, Alan Davis
- *Code Complete*, Steve McConnell
- *Applying UML & Patterns 2nd Ed*, Craig Larman
- *Programming Pearls 2nd Edition*, Jon Bentley
- *Refactoring*, Martin Fowler
- *Writing Effective Use Cases*, Cockburn
- *Object Oriented Analysis and Design*, Grady Booch
- *Design Patterns*, Erich Gamma et al
- *Conceptual Blockbusting*, James Adams
- "The Humble Programmer," Edsger Dijkstra

Tester

Knowledge and Skills

In Construx's judgment, software testers need the following set of knowledge and skills to perform their roles effectively:

- Estimation Techniques
- Peer Review Methods
- Black Box Testing Fundamentals
- Test Case Preparation
- Equivalence Classes
- Coverage Levels
- Written and Verbal Communication
- Interpersonal Skills

Professional Development Plan

The following sections describe the experience, training, and reading that will support development of the knowledge and skills necessary to perform effectively as a software tester.

Experience

- Act as a reviewer of a test plan
- Participate in an informal review
- Participate in a formal inspection
- Perform testing on at least one project
- Create test cases on a project

- Write two or more Software Test Plans
- Act as Quality Lead on at least one medium sized project

Training

- Inspections
- Black Box Testing Fundamentals
- Peer Reviews
- Real World Software Testing
- Success Through Risk Management
- Pass *Certified Software Test Engineer* exam

Reading

- "Software Engineering Code of Ethics and Professionalism," ACM/IEEE-CS
- *Software Project Survival Guide,* Steve McConnell
- *Mastering the Requirements Process*, Robertson and Robertson
- *Mythical Man-Month*, Fred Brooks
- *Peopleware*, DeMarco and Lister
- *Rapid Development,* Steve McConnell
- *201 Principles of Software Development*, Alan Davis Testing Computer Software, Cem Kaner et all
- *The Art of Software Testing,* Glenford Myers
- *Complete Guide to Software Testing,* Bill Hetzel
- *Software Inspection*, Tom Gilb and Dorothy Graham
- *Metrics and Models in Software Quality Engineering*, Stephen H Kan
- *Managing the Software Process*, Watts Humphrey
- *Quality is Free,* Crosby
- "Design and Code Inspections to Reduce Errors in Programs," Michael E. Fagan
- "Seven Deadly Sins of Software Reviews," Karl Wiegers

5. Construx Course Catalog

Recommended Construx Courses By Function/Job Title

Construx offers a wide variety of software development seminars. Depending on an individual's primary functions, specific seminars will be the most beneficial in improving job performance.

	Construx Seminars													
Job Function/ Title	**Black-Box Testing Fundamentals**	**Building Better Software with Use Cases**	**Effective Software Project Management**	**Inspections**	**Keys to Successful Project Outsourcing**	**Object Oriented Analysis and Design, using the UML**	**Peer Reviews for Higher Quality and Productivity**	**Rapid Development: Accellerating Time to Market**	**Real World Requirements**	**Real World Software Testing**	**Software Estimation: In-Depth Workshop**	**Software Project Survival: Planning for Success**	**Steve McConnell's Code Complete**	**Success Through Risk Management**
Analyst		C		C	R	C	C	R	C	R	R			R
Architect		R		C		C	C	R	C		R			R
Designer, Programmer, Tool Smith		R		R		C	C		C	C	R		C	R
Quality Assurance	C	R		C	R		C	R	R	C	R	R		C
User-interface Designer		R		R		C	C		C					R
Project Manager	R	R	C	C	R		C	R	C		C	C		C
Product Manager		C	R	R	C	R	C	R	C		C	R		C
End-User Liaison	R	R					C		R	R				
Tester	C	R		C			C		R	C	R			
Build Coordinator	R						R						C	
End-user documentation		C		C		R	R		R					R

C = Core
R = Recommended
Construx can assist you in developing a comprehensive curriculum that includes seminars other than Construx's offerings through our Training Needs Assessment and Professional Development Ladder services.

Black-Box Testing Fundamentals

(1 PDU)

Overview

How do you test software when you didn't write the code? Testing is an important activity throughout the lifecycle of a software development project. Understanding how to plan for black-box testing is critical to the success of the testing effort, and therefore the success of your project. This course focuses on practical techniques for planning and executing black-box testing.

This Course Includes The Following Topics:

Test Planning

- When should you plan for testing?
- How much testing is enough?
- How can you make your testing more effective?
- How can you make your testing more efficient?
- How does testing relate to other software development activities?
- How do you evaluate your test case designs?
- How should you handle regression testing?

Domain Testing

- Where can you find domain specifications?
- How do you analyze domain specifications?
- What common domain defects should you test for?
- How do you design input domain tests?
- How do you design output domain tests?
- How do you analyze domain boundaries?

Testing from Requirements

- In what form are your requirements specified?
- How can you create test cases from use cases?
- How can you create test cases from relationship models?
- How can you create test cases from state-transition models?

Tool Support for Testing

- What tools are useful for supporting testing activities?
- How can automation improve the efficiency of your testing?

Testing the Tests

- How do you know your tests are reliable?

- How do you know your tests are effective?
- How can you reuse test case designs?
- How do you uncover and improve low quality tests?

Benefits of Attending

This workshop will help attendees learn about practical techniques for planning, designing, and executing effective black-box testing on real software projects. Participants will learn to determine how much testing is enough for your project, whether your test cases are adequate, and how to minimize wasted testing effort.

Who Should Attend

This seminar will be useful to testers, test leads, quality assurance professionals, and quality assurance leads.

PMI Knowledge Areas Covered

Quality Management

Building Better Software With Use Cases

Overview

Use cases are a powerful technique for gathering, organizing and verifying users' business-systems requirements. In this two-day seminar, leading software expert Meilir Page-Jones will show you what use cases are and how to use them to understand, model, and validate user requirements quickly and precisely. He will explain how to distinguish a good use case from a bad one. He will delve into the highly effective technique of business-event modeling and explain how a thorough understanding of business events simplifies creating sound use cases. Meilir will also present a use-case-based project-matrix technique for accurately planning and tracking projects.

This Course Includes The Following Topics:

Basic Use Cases

- Definition of "use case"
- The system in the business context
- Whose business reality is it, anyway?
- Actors, people, jobs, and roles: How do they interrelate?
- Three forms of Use-Case Goal

Practical Tips for Developing Use Cases

- Five ways to organize a use-case list: Which one is best?
- How use cases relate to one another
- Defining use cases
- The dimensionality of events and use cases
- Use-case preconditions and post conditions
- Defining and documenting use cases

Applying the Strength of Business Events to Your Software Projects

- Five criteria for a legitimate business event
- Event detectors versus event reporters
- Four kinds of business events
- How business events yield use cases

Realizing the Project-Wide Benefits of Use Cases

- How to make the most of use cases throughout the project
- Use cases and the class/data model
- Using use cases to model the user interface
- Using use cases to analyze response time and distributed system architectures
- Designing, programming, testing, delivering, and managing via use cases

Benefits of Attending

Attendees will gain an in-depth understanding of use cases. They will learn the *full* benefits of use cases including requirements gathering, user-interface prototyping and design, system-architecture design, detailed software design, programming, and testing. Attendees will learn practical ways to define use cases both distinctly and succinctly. They will practice designing and developing use cases. And they will have ample opportunities to discuss their specific use-case questions with Meilir Page-Jones.

Who Should Attend

This seminar will be useful to requirements analysts, users, system designers, and project-team leaders and managers.

Steve McConnell's Code Complete

Overview

What if you could write better code? What if you could write code faster? Learn dozens of time-tested tips and techniques from the coding guru who wrote the programming classic, *Code Complete*. This intense, one-day workshop is based on Steve McConnell's best-selling programming book, *Code Complete*, a computing industry classic that won the Jolt Excellence award for best programming book of the year and has been translated into ten languages. Steve will also present ways that software construction practices have advanced since the book was published.

This Course Includes The Following Topics:

Introduction

- Construction's critical role in software development
- The stable core of programming knowledge

Defensive Programming Practices

- Anticipating change
- Sources of change
- Designing families of programs
- PDL-to-code process
- Effective approaches to error processing
- A winning strategy for code optimization

Techniques for Managing Design Complexity

- Abstraction
- Encapsulation
- Modularization
- Information hiding
- Coupling
- Cohesion
- Design patterns
- Separation of concerns
- Stepwise refinement

Creating Effective Variable and Function Names

- *Coding horror:* Ineffective names
- Status variables
- Loop indices

- "Temporary" variables
- Function names
- General principles

Creating High-Quality Functions

- *Coding horror*: A low-quality routine
- Valid and invalid reasons to create a routine
- Program layout techniques
- Low-effort, high-payoff commenting techniques

Quality Practices

- *Coding horror*: Debugging by superstition
- Upstream/downstream costs
- Psyching out defects
- A scientific approach to debugging
- Tips for finding defects
- Tips for fixing defects
- Technical reviews
- Unit testing

Emerging Issues in Professional Programming

- Teamwork
- Cooperation
- Communications
- The new profession of software engineering

Benefits of Attending

Attendees will learn effective software construction practices including techniques that shorten development time, reduce errors, make debugging easier, and produce code that is easier to maintain. Readers of *Code Complete* will hear about ways software construction practices have advanced since the book's publication.

Who Should Attend

This seminar will be useful to software engineers, developers, and anyone who wants to learn best coding practices.

Effective Software Project Management

(12 PDUs)

Overview

Knowledge of effective software management practices might not be widespread, but leading companies have known effective management techniques for years. In this hands-on course, Meilir Page-Jones will teach you how to deal with seemingly impossible deadlines, reduce your project costs, improve your software quality, and improve your own management abilities! Manage your team so that you get the best quality, productivity, and motivation—and keep team morale high.

This Course Includes The Following Topics:

- Why software project management is different from plain old management
- Aligning projects with overall corporate objectives
- Relating project costs and benefits
- The five vital activities of project management
- Project management strategies for different situations
- The project manager as director and producer
- Quality and productivity—how to measure and improve them
- Effective estimating techniques
- The truth about time pressures and deadlines
- The seven reasons software methodologies fail—and how to avoid them
- Worthwhile project reporting techniques
- Conducting productive project reviews and walkthroughs
- Keys to an effective work environment
- The concept of the mediocrity and how to escape it
- Cultivating the qualities and attitudes of a successful project manager

Benefits of Attending

The workshop will include case studies and project vignettes drawn from real projects through which attendees will be able to practice project-management techniques. Attendees will have time to discuss specific issues with the instructors. Class size is limited!

Who Should Attend

This seminar will be useful to current project managers and team leads as well as senior software engineers and senior software developers who are about to take on the role of project manager.

PMI Knowledge Areas Covered

Integration Management, Scope Management, Time Management, Cost Management, Quality Management, Human Resource Management, Communication Management and Risk Management

Object-Oriented Requirements Analysis & Design Using The UML

(4 PDUs)

Overview

This seminar provides a programming language-independent overview of object-oriented requirements analysis and design using the UML. Special attention is given to relating OOA and OOD concepts to real-world software. No prior knowledge of object-oriented development is required.

This Course Includes The Following Topics:

Analysis v. Design, What's the Difference?

- Requirements and non-requirements
- Essential requirements and design requirements
- Separating essential and design requirements
- Defining analysis and design

What Does it Mean to be Object-Oriented?

- A brief history of object-orientation
- Comparing and contrasting structured and object-oriented
- Software object -- from concept to code
- Defining object-orientation

Object-Orientation Meets Analysis

- Use-case diagrams
- Class diagrams
- Collaboration diagrams
- Sequence diagrams
- State diagrams, actions, and activities

Object-Orientation Meets Design

- The transition to design
- Designing interfaces
- High-level software design
- Low-level software design
- Software optimization

What's Next With The UML?

Benefits of Attending

Attendees will gain an in-depth understanding of how to improve their object-oriented project's requirements analysis and design and an introduction to the UML. They will have time to discuss their specific issues with Steve Tockey. They will practice techniques that they can apply to their own projects when they return to work.

Who Should Attend

This workshop will be useful to programmers, designers, analysts, and technical leads.

PMI Knowledge Areas Covered

Scope Management

Peer Reviews For Higher Quality And Productivity

(4 PDUs)

Overview

Upstream inspections can eliminate up to 90 percent of a software project's defects while simultaneously producing net cost and schedule savings of 10 to 30 percent. That is why Peer Reviews are a critical component of any effort to reduce software defects or project costs and achieve the shortest possible schedule. Used skillfully, Peer Reviews may be one of the best investments your organization makes in the improvement of its software processes.

Attendees will see how Peer Reviews allow developers to share technical expertise and project knowledge, as well as improve overall development skills. They will also gain an understanding of how Peer Reviews provide software project managers with greater assurance that a software engineering process is being followed and that status tracking is fact- as opposed to wish-based.

This Course Includes The Following Topics:

The Costs of Software Defects

Forms of Peer Review

- Desk checks
- Walkthroughs
- The walkthrough process
- Formal inspections
- Advantages and disadvantages of each type of peer review
- Software previews

Formal Inspections

- Defined roles for participants
- Formal inspection process
- Use of checklists
- How to build checklists
- Variations on the formal inspection process

Formal Inspections with Data Collection

- What to measure
- Why to measure

Formal Inspections with Measurement-Based Improvement (aka Fagan Inspections)

- Improving the inspection process
- Improving the software process

- Pareto analysis and causal analysis
- An introduction to statistical process control

Guidelines for Successful Reviews

- Why peer reviews can fail and what you can do to maximize success
- Establishing a peer review program

Additional Resources

Benefits of Attending

Attendees will learn each of the different forms of peer review, how to select the most appropriate form of peer review, and how to apply that selected form of peer review.

Who Should Attend

This seminar will be useful to lead software developers, lead software engineers, software managers, and team members.

PMI Knowledge Areas Covered

Quality Management

Rapid Development: Accelerating Time To Market

(12 PDUs)

Overview

Time to market is a constant issue for software projects. This seminar presents specific, practical techniques for dramatically improving time to market on real-world software projects. Case studies based on high-profile projects and projects drawn from Construx's consulting experience are interspersed throughout the workshop to illustrate development principles and give students experience applying best practices in rapid development.

This seminar is based on Steve McConnell's best-selling book, *Rapid Development* (Microsoft Press, 1996).

This Course Includes The Following Topics:

Core Issues in Rapid Development

- Mental models
- Kinds of schedule-oriented practices
- Real-world tradeoffs
- General strategy for attaining rapid development

Special Core Issue: Quality and Rapid Development

- Relationship between quality and development speed
- When quality costs more
- When quality costs less
- Role of quality on rapid development projects

Achieving Rapid Development

- Rapid development strategy
- Avoiding classic mistakes
- Applying software fundamentals
- Active risk management

Avoiding Classic Mistakes

- Most common classic mistakes
- Catalog of classic mistakes
- Avoiding classic mistakes

Applying Software Fundamentals

- Technical fundamentals

- Technical management fundamentals
- Quality assurance fundamentals

Active Risk Management

- Risk identification
- Risk analysis
- Risk prioritization
- Risk control

Lifecycle Selection

- Matching lifecycles with projects' schedule needs
- Staged delivery
- Design to schedule
- Design to tools
- Spiral model
- Evolutionary prototyping
- Evolutionary delivery

Human Factors

- Senior staff
- Motivation
- Productivity environments
- Rewards and incentives

Rapid Development Best Practices

- Scheduling
- Strategic technology adoption
- Feature-set control

Synergy

Benefits of Attending

Attendees will gain an in-depth understanding of the major issues that affect software development speed. Through numerous case studies, they will practice rapid development techniques that they can apply to their own projects. They will have time to discuss their specific issues with the instructor. And they will develop a back-to-work action plan.

Who Should Attend

This course will be useful to technical leads and managers, QA leads and managers, and product managers.

PMI Knowledge Areas Covered

Integration Management, Scope Management, Time Management, Cost Management, Quality Management, Human Resource Management and Risk Management

Real World Requirements

(4 PDUs)

Overview

The course presents practical techniques for exploring user needs, capturing requirements, controlling changes, and building highly satisfactory software. It explains how leading edge companies use requirements engineering to support successful software projects. Through a mix of lecture, discussion, and exercises, attendees will learn many useful requirements engineering practices.

This Course Includes The Following Topics:

Why Good Requirements Matter

The Requirements Process

- Components of the requirements process
- Requirements in the context of the overall project lifecycle

A Definition of High-Quality Requirements

- Kinds of requirements
- Facets of a high-quality requirement
- Requirements tags and requirements traceability
- "Fit criteria"
- Prioritizing requirements
- Quality criteria for requirements

Special Topic: Ambiguity

- Why ambiguous requirements are a serious problem
- Sources of requirements ambiguity
- Recognizing ambiguity in requirements
- Reducing ambiguity in requirements

Eliciting Requirements

- The importance of project vision statements
- Identifying the right people: clients, customers, and users
- Getting the right people involved in the requirements effort
- Drawing requirements out of people
- Other useful techniques: (JAD/JRP, apprenticing, market surveys, focus groups, etc)
- Other sources of requirements

Specification Techniques (aka "Packaging the Requirements")

- Textual specifications including IEEE Std 830-1998
- User manuals as specifications
- Models as specifications
- Prototypes as specifications

Requirements Processes

- A tailorable requirements process
- Tailoring guidelines
- Requirements and software testing

Requirements Management

- Feature interaction and its effect on complexity
- Requirements scrubbing
- Requirements change management
- Matching the product to the resources

Leading Edge Practice: Product Family Techniques

Benefits of Attending

Attendees will learn practical techniques for gathering, capturing, and controlling changes to requirements. Attendees will practice many of the techniques under the instructor's guidance and have their personal questions answered by a leading expert in software engineering.

Who Should Attend

Software developers, software engineers, managers, requirements analysts--anyone engaged in gathering, documenting, analyzing, or managing customer requirements for software applications.

PMI Knowledge Areas Covered

Scope Management

Real World Software Testing

Overview

Software testing is an important part of any software quality assurance program. Effective testing can be the difference between low defect software that is highly satisfying to your users and buggy software that gives rise to a never-ending stream of user complaints.

This Course Includes The Following Topics:

Introduction

- Kinds of defects
- Testing v. debugging
- Economies of testing
- Impediments to testing

Generic View of Testing

- Test planning
- Test case design
- Characteristics of a good test case
- Test execution
- Regression testing

Testing and the Software Lifecycle

- Generic software lifecycle
- Unit testing (stubs and drivers, automating)
- Integration and component testing (bottom-up, top-down, umbrella)
- Object-oriented software testing
- System Testing
- Acceptance testing

Functional Testing (Black-Box)

- Requirements testing
- Grammar (syntax) testing

Functional Coverage Criteria

- Requirements coverage
- Input domain coverage
- Output domain coverage

Structural Testing (White-box)

- Software as a graph

- Control-flow testing
- Data-flow testing

Structural Coverage Criteria

- Statement coverage
- Decision (branch) coverage
- Modified condition/decision coverage
- All definitions (reach) coverage
- All uses coverage
- All DU paths coverage

Testing from Contemporary Specifications

Tool Support for Testing

Testing the Tests

Additional Resources

Benefits of Attending

This workshop will help attendees learn the different forms of software testing and test coverage criteria. Specifically, you will learn how to develop optimal test cases--both functional (black-box) and structural (white-box)--and then how to plan and carry out effective and efficient testing on real software projects. Participants will gain a better understanding of why testing can fail, as well as learn to decide how much testing is enough. You will also discover tips to make your software more testable.

Who Should Attend

This seminar will be useful to software leads, software engineers, quality assurance leads, quality engineers.

Software Estimation: In-Depth Workshop

(12 PDUs)

Overview

This course focuses on providing many useful rules of thumb and procedures for creating software estimates ("the art of estimation"), and provides a brief introduction to mathematical approaches to creating software project estimates ("the science of estimation"). This course provides techniques for making sure estimation is treated as an analytical rather than a political process. It explains how to negotiate effectively with other project stakeholders (such as marketing, management, and your clients) so that everyone "wins." This course features extensive lab work to give you hands-on experience creating many different kinds of software estimates.

This Course Includes The Following Topics:

Estimation Background

- Benefits of accurate estimation
- Disadvantages of accurate estimation
- Estimates versus targets
- Macro versus micro estimation
- Estimation "art" versus estimation "science"
- State of the art and limits on estimation accuracy

Basic Software Estimation

- Basic steps in creating a software estimate
- Estimate refinement over the course of a project
- Change in estimation approaches over the course of a project
- Standardized estimating procedures

Popular Estimation Methodologies

- Cocomo II
- Putnam's Method

Scope Estimation

- Estimation by analogy
- Estimation by expert judgment
- Estimation by module breakdown
- Estimation by function points

Effort Estimation

Schedule Estimation

Special Topics in Software Estimation

- Estimating individuals' work
- Estimating small-team projects
- Estimating maintenance projects
- Calibrating an estimate for your team and organization

Automated Estimation Support

- Product demos
- Calibration modes: project type, cost drivers, and historical data
- Finding the optimum solution
- Determining estimate quality

Human Roles in Estimation

- How to present and explain an estimate
- Estimation and negotiation

Benefits of Attending

You will learn how to calibrate your estimates to be accurate for your specific development environment. You will learn effective techniques for estimating both large and small projects.

Who Should Attend

This seminar will be useful to software engineers, developers, and anyone who wants to learn to effectively estimate software costs and schedules for their environments.

Software Project Survival For Business Leaders

Overview

The success or failure of a software project often hinges on factors that are external to the project team itself. Is the project team given a sufficient set of precise requirements or only a few sketchy *desirements*? Are the requirements stable over the life of the project, or do they change more often than the weather? Is the software team on good terms with the customer, or is it an antagonistic relationship? And so on.

This seminar is for people who are not members of the immediate software project team but who are in a position to have a significant influence on the success or failure of the project. This includes middle- and upper-management, sales people, product (as opposed to project) managers, etc. In this seminar you will learn how seemingly innocuous decisions and actions on your part can have a major impact on the success of software projects. You will also find what kinds of things a well-managed and well-run software project can do to increase its likelihood of success.

This seminar is based on Steve McConnell's best-selling book, *Software Project Survival Guide* (Microsoft Press, 1998).

This Course Includes The Following Topics:

- Causes of Project Difficulty
- Software Project Survival Concepts
- Software Survival Fundamentals
- *Key Survival Skill*: Planning Checkpoint Reviews (feasibility studies)
- Key Survival Skill: Change Control
- Software Best Practices

Case studies are interspersed throughout the workshop to illustrate how the principles apply to a variety of real world situations. These case studies are ripped from headlines in the *Wall Street Journal*, *Forbes*, *and Business Wee*k as well as the experiences of our consultants.

Benefits of Attending

Attendees will gain an in-depth understanding of the major issues that affect software project success. They will have time to discuss their specific issues with Steve Tockey.

Who Should Attend

This seminar will be useful to those who are not immediate members of a software project team, who are in a position to have a significant influence on the success or failure of a project, and who desire project survival.

Software Project Survival Guide: In-Depth Workshop

(12 PDUs)

Overview

Many people in the software industry are thrust into positions of responsibility for software project outcomes, but few are given formal or informal training in how to make that happen. In this presentation, the workshop leader describes specific steps that small and medium-sized development teams should take to succeed on projects from six to 12 months long. This Software Survival Process is based both on the instructor's direct experiences on software projects and on experience gained from the projects reviewed in Construx Software's project assessment and project recovery consulting business.

This seminar is based on Steve McConnell's best-selling book *Software Project Survival Guide* (Microsoft Press, 1998).

This Course Includes The Following Topics:

- Causes of project failure
- Successful and unsuccessful project recovery techniques
- Survival concepts
- Role of process in software project survival
- Survival fundamentals
- Key survival skill: Change control
- *Key survival skill*: Risk management
- *Key survival skill*: Project tracking
- Planning checkpoint reviews (feasibility studies)
- Back-to-work action plan

Case studies are interspersed throughout the workshop to illustrate how survival principles apply to a variety of real world situations. These case studies are ripped from headlines in the *Wall Street Journal*, *Forbes*, and *Business Week*, as well as drawn from Construx Software's own experience.

Benefits of Attending

Attendees will gain an in-depth understanding of the major issues that affect software project success. They will have time to discuss their specific issues with the instructor. They will practice techniques that they can apply to their own projects when they return to work, and they will develop a back-to-work action plan.

Who Should Attend

This seminar will be useful to software engineers, developers, and anyone who wants to learn software project survival.

Software Project Survival Guide: Planning For Success

(12 PDUs)

Overview

Many people in the software industry are thrust into positions of responsibility for software project outcomes, but few are given formal or informal training in how to make that happen. In this workshop, you will learn specific steps that small and medium-sized development teams should take to succeed on projects from 3 to 12 months long, with extensive use of software best practices. This Software Survival Process is based on the instructor's direct experiences on software projects and on experience gained in Construx's consulting and coaching business.

This seminar is based on Steve McConnell's best-selling book *Software Project Survival Guide* (Microsoft Press, 1998).

This Course Includes The Following Topics:

Introduction to Software Project Survival

- Software project survival test
- Causes for project failure
- Key survival concepts

Project Planning

- Profile of a successful project
- Contents and structure of a project plan
- *Best Practice*: Chartering the project for success
- Working with constraints
- Identifying risks and leveraging assets
- Organizing and staffing the project

Project Estimation

- Cone of uncertainty
- Targets vs. estimates
- Estimation sanity checks
- Refining the estimate
- *Best Practice:* Planning checkpoint reviews

Managing Requirements

- Characteristics of good requirements
- Effective requirements practices in a nutshell
- *Best Practice:* User interface prototyping

Quality Assurance Planning

- Economics of software quality
- *Best Practice:* Upstream defect detection
- Best Practice: Peer reviews
- *Best Practice:* Daily build and smoke test

Tracking and Controlling the Project

- Configuration management
- *Best Practice:* Change control
- Tracking techniques: Status reports, milestones, and better alternatives

Risk Management

- Risk identification, analysis, prioritization, and control
- *Best Practice:* Top 10 risks list
- *Best Practice:* Risk management plans

Building on Success

- Selling process to your organization
- *Best Practice:* Software project log book
- *Best Practice:* Effective post mortem reviews

The extensive labs used throughout the workshop focus on developing a project plan for your specific project. Case studies are interspersed throughout the workshop to illustrate how survival principles apply to a variety of real-world situations.

Benefits of Attending

Attendees will gain an in-depth understanding of the major issues that affect software project success. They will be able to develop a project plan that can be used on their current project. They will have time to discuss their specific issues with the instructor. They will practice techniques that they can apply to their own projects when they return to work.

Who Should Attend

This seminar will be useful to software engineers, developers, and testers placed in technical leadership roles. It is also useful to experienced managers who are new to software, and anyone who wants an introduction to software project best practices.

PMI Knowledge Areas Covered

Integration Management, Scope Management, Time Management, Cost Management, Quality Management, Human Resource Management and Risk Management

Success Through Risk Management

(12 PDUs)

Overview

This course focuses on practical techniques you can use to identify, analyze, prioritize, and control risks and on effective strategies to manage the most common risks software projects face.

This Course Includes The Following Topics:

Introduction

- A definition of risk
- What is risk management?
- The need for risk management

Risks in Detail

- The scope of risks
- Risks as cause-effect
- Ultimate causes and ultimate effects
- Risk timeframes
- Assets

Risk Identification

- Categories of risks
- Common project risks
  -Inadequate requirements
  -Creeping requirements
  -Gold plating
  -Overly optimistic schedule
  -High turnover
- Practical techniques to identify risks and assets

Risk Analysis/Prioritization

- Risk probabilities
- Risk severities
- Techniques for accurately estimating risk probabilities
- Determining risk exposure
- Prioritizing risks
- Analyzing/prioritizing assets

Risk Response Planning

- Risk response strategies
  -Prevent
  -Mitigate
  -Transfer/share
  -Contingency plan
  -Risk reserve/provision
  -Ignore
- Planning risk response

Risk Responses

- Responses to requirements problem
- Responses to inadequate project management
- Responses to inattention to upstream quality
- Responses to misunderstood project dynamics
- Strategies for maximizing common assets

Risk Response Control

- Response control through project tracking
- Ongoing risk reassessment

Summary

Benefits of Attending

Attendees will gain a deep understanding of common project risks. They will learn specific remediation strategies and techniques that will maximize their chance of success. They will learn the key practices to actively attack, prevent, and eliminate software project risks.

Who Should Attend

This workshop will be useful to project managers, program managers, technical leads, QA leads, and team leads.

PMI Knowledge Areas Covered

Risk Management

*Construx is a recognized provider registered with the PMI Registered Educational Provider Program (PMI R.E.P.).

6. How Construx Can Help

Our professional software development program is designed to meet your organization's specific needs. It increases your software team's technical skills, judgment, productivity and overall contribution to software development best practices by specifying the knowledge and experience essential for success in your organization.

Training Needs Assessment

Our consultants follow a structured process to quickly and accurately assess the areas in which training will provide the greatest benefit to your organization.

Corporate Ladder

We work with your organization to tailor the Construx Professional Development Ladder to meet the needs of your organization. Common tailoring includes consolidation or additions of knowledge areas, modifications to employee progression paths, and modification of specific requirements.

Career Path Guidance

To motivate and guide your employee's development, we provide specific career paths based on your corporate needs. This guide provides a summary of the knowledge areas and skills progression for each technical role.

Professional Development Plan Template

This template is designed to jump start each employee's adoption of the ladder. This can be found on Construx's website at www.construx.com/resources/pdl/sampleplans/.

Mentor/Manager Handbook

To ensure your managers and/or mentors can support employee development, we provide a Handbook that outlines how employees progress up the ladder, how the ladder and performance reviews interact, and what ongoing professional development activities are supported by the organization.

Interview Practices

By providing practices, techniques, and specific interview questions based on the ladder, you can improve your interviewing and hiring practices.

Performance Review Program

We provide templates and coaching on how to link the professional development program to your current performance review program or establish a performance review program based on your ladder.

7. About Construx

Construx Software was founded in 1996 to provide a new level of practical, real-world support to the software community. Most training and consulting companies don't build software themselves. Most software-building companies don't make adequate use of training or consulting for their own projects.

Construx Software provides industry leading

- Consulting
- Training
- Software engineering products
- Outsourced software projects

By focusing on all four areas, we provide our customers with the latest state-of-the-art practices, proven on real projects.

Construx Software Professionals

Construx software professionals tend to be much more experienced than their counterparts in other companies: the typical Construx developer has more than 10 years of software development experience. Many have taught classes in their fields including the use of particular programming languages, software design, and software project management. Some have published books or articles. Many have attained advanced degrees in software engineering, computer science, and related fields.

Construx personnel have received some of the highest honors the software industry has to award and have participated in delivery of award-winning products and many premiere horizontal market and vertical market programs.

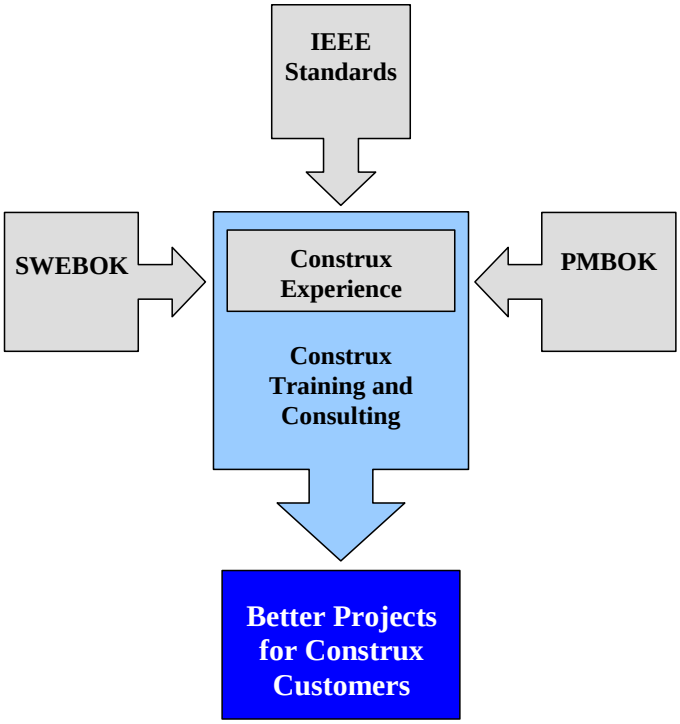
Steve McConnell, CEO/Chief Software Engineer

Steve McConnell is one of the world's foremost experts on software development and software engineering. In 1998, readers of *Software Development* magazine voted him one of the three most influential people in the software industry, along with Bill Gates and Linus Torvalds. Steve is the author of *Code Complete* and *Rapid Development*, winners of *Software Development*'s Jolt award for best software development book of the year in 1994 and 1997. Steve is the author of the best selling *Software Project Survival Guide* (1998) and *After the Gold Rush: Creating a True Profession of Software Engineering* (1999). Steve also serves as the editor-in-chief of *IEEE Software* magazine.

Construx and Software Development Standards

Construx incorporates the work of standards bodies in everything we do. We distill the information gathered from such sources as the Software Engineering Institute (SEI), ISO Standards, IEEE Software Engineering Standards, Software Engineering Body of Knowledge (SWEBOK), Object Management Group (OMG), Project Management Body of Knowledge (PMBOK), and other sources.

This refinement and focusing of the general knowledge into specific practices, knowledge areas, and experience gives you the benefit and transferability of the standards applied to your domain.



Construx is a Recognized Education Provider for the PMI, sponsor of the SWEBOK, a member of the OMG, a member of the IEEE Computer Society's Professional Practices Committee, on the board of many IEEE Computer task forces, and active in many other local, national, and international initiatives. By staying active in standards activities, Construx ensures that we are always bringing the latest and best practices to our clients.

Construx applies the knowledge in real projects both internally and for our clients so we know what works. Construx has done the detailed work of implementing best practices and can share our success with you.

IEEE-CS

The IEEE-CS (Institute of Electric and Electrical Engineers Computer Society) is the leading professional organization for practicing software developers. The IEEE defines and publishes software engineering standards, and the IEEE-CS has been on the forefront of professionalism issues including defining the software engineering body of knowledge (SWEBOK), defining university curriculums in software engineering, accrediting software developers, and sponsoring many leading conferences and publications.

Construx has been active in the IEEE in numerous ways. Construx's CEO and Chief Software Engineer, Steve McConnell, was Editor in Chief of *IEEE Software* magazine from 1998-2002. McConnell also served as one of the charter members of the IEEE-CS's Professional Practices Committee, which oversees the IEEE-CS's software engineering professionalism activities. Other Construx engineers have participated in development of IEEE software engineering certification programs and curriculum guidelines. Nearly all of Construx's software engineers have received their IEEE Certified Software Development Professional (CSDP) accreditation.

The SWEBOK

Achieving consensus by the software engineering profession on a core body of knowledge is a key milestone in all disciplines. The Guide to the Software Engineering Body of Knowledge project is an initiative to reach this consensus.

In other engineering disciplines, the accreditation of university curricula and the licensing and certification of practicing professionals are taken very seriously. These activities are seen as critical to the constant upgrading of professionals and, hence, the improvement of the level of professional practice. Recognizing a core body of knowledge is pivotal to the development and accreditation of university curricula and the licensing and certification of professionals.

The software engineering body of knowledge is an all-inclusive term that describes the sum of knowledge within the profession of software engineering. Since it is usually not possible to put the full body of knowledge of even an emerging discipline, such as software engineering, into a single document, there is a need for a Guide to the Software Engineering Body of Knowledge. The Guide will seek to identify and describe that subset of the body of knowledge that is generally accepted, even though software engineers must be knowledgeable not only in software engineering, but also, of course, in other related disciplines.

About PMI & The PMBOK

The Project Management Institute in its Guide to the Project Management Body of Knowledge (PMBOK) defines "generally accepted" knowledge for project management in the following manner:

"Generally accepted" means that the knowledge and practices described are applicable to most projects most of the time, and that there is widespread consensus about their value and usefulness.

"Generally accepted" does not mean that the knowledge and practices described are or should be applied uniformly on all projects; the project management team is always responsible for determining what is appropriate for any given project.

The Guide to the Project Management Body of Knowledge is now an IEEE Standard.

Since its founding in 1969, Project Management Institute (PMI) has grown to be the organization of choice for project management professionalism. With almost 90,000 members worldwide, PMI is the leading nonprofit professional association in the area of Project Management. PMI establishes Project Management standards, provides seminars, educational programs.

In the PMBOK, the Project Management Institute (PMI) describes in detail the project management processes that apply to most projects, most of the time. PMI does not offer details on product-oriented processes. Instead, it states that product-oriented processes are typically defined by the project lifecycle and vary by application area.

Other Resources

The Software Engineering Laboratory (SEL) at NASA's Goddard Space Flight Center is doing some of the most interesting work in software development today. The SEL was awarded the IEEE's first award for achievement in software processes. This site contains the full text of the *Software Measurement Guidebook* and other resources.

Software Engineering Institute (SEI). The SEI is responsible for the capability maturity model (CMM) and other valuable -- and sometimes controversial -- developments. They're doing a lot of exciting work. Many of the SEI documents are available in Adobe Acrobat (PDF) and Postscript formats from this site.

Software Project Manager's Network. This site is the home base of the DoD's Best Practices Initiative. It includes *The Book of Software Management Questions* (The Yellow Book), *The Program Manager's Guide to Software Acquisition Best Practices* (The Big Book), and *The Condensed Guide to Software Acquisition Best Practices* (The Black Book).

Software Assurance Technology Center. This is another software site associated with NASA's Goddard Space Flight Center. It focuses on software quality.

Software Productivity Consortium This is a DoD sponsored consortium aimed at improving software productivity. "Military Intelligence" might be an oxymoron, but the DoD spends more money on software development than any other single organization, and some of what they've come up with is worth a close look.

The *Software QA and Testing Resource Center* includes FAQ's, software QA and testing resources, software QA and test tools, web site and site management tools, and a "bookstore."

*IEEE Computer Society***.** A professional membership society for software developers, the IEEE Computer society is responsible for some of the best publications in the industry. Their publica-

tions include *Computer*, *IEEE Transactions on Software Engineering*, and what many consider to be the best software development periodical available, *IEEE Software*.

Association of Computing Machinery (ACM). This is another professional membership society for computer professionals. It was the first computing society and publishes the *Communications of the ACM*.