

Got Software?

What Managers And Engineers Need To Know^{1,2}

P. A. "Trisha" Jansma
Jet Propulsion Laboratory, California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
818-354-0647
Patti.A.Jansma@jpl.nasa.gov

Abstract—As part of a JPL-wide software quality initiative aimed at addressing the challenges of developing, managing and acquiring software, a team at JPL generated a detailed Software Training Plan for training both managers and engineers. The team took the approach of treating the software training program as though it were a system development task, and went through all the typical phases of system development including requirements, design, implementation, etc.

During the requirements collection phase, the team conducted dozens of interviews and identified the specific skills needed. The skills fell into categories such as software management, software engineering, systems engineering and other technical areas. However, an equally important finding was that several "soft" skills were deemed critical for the successful and timely management and implementation of software-intensive systems. This paper discusses JPL's approach and "lessons learned" from planning and delivering a software training program in an engineering and scientific environment.

TABLE OF CONTENTS

.....	
1. INTRODUCTION	1
2. SOFTWARE TRAINING CUSTOMERS	2
3. SOFTWARE TRAINING REQUIREMENTS	2
4. SOFTWARE TRAINING GOALS	3
5. SOFTWARE TRAINING PROCESS	3
6. SOFTWARE TRAINING PROVIDERS	4
7. SOFTWARE TRAINING COURSES	4
8. MANAGERIAL TRAINING COURSES	6
9. TRAINING IMPLEMENTATION	6
10. JPL LESSONS LEARNED	7
11. CONCLUSIONS	8
12. ACKNOWLEDGEMENTS	8
REFERENCES	8
BIOGRAPHY	9

1. INTRODUCTION

About JPL

The Jet Propulsion Laboratory (JPL), located in Pasadena, California is a non-profit federally funded research and development center (FFRDC) which is operated under contract by the California Institute of Technology (Caltech) for the National Aeronautics and Space Administration (NASA). JPL is part of the U.S. aerospace industry, and is NASA's lead center for robotic exploration of the solar system. In addition to its work for NASA, JPL conducts tasks for a variety of other federal agencies, such as the Department of Defense, the Department of Transportation, the Department of Energy, etc. JPL has approximately 5500 employees: 4500 in the technical and programmatic divisions and 1000 in the administrative divisions. Its annual budget is approximately \$1.4 billion.

Background

Motivated by some recent, highly visible failures in which software was implicated in mission loss (e.g., Mars '98) and by a NASA-wide software engineering initiative, JPL has recently begun a major software quality improvement effort. The Software Quality Improvement (SQI) Project was created in FY 2002 to establish an on-going operational program that results in the continuous, measurable improvement of software quality at JPL. The SQI Project is chartered to provide education, training, mentoring, and consulting for projects and practitioners in order to enable and promote software best practices, and to leverage JPL experience in software engineering in support of major software projects, throughout the entire software life-cycle.

As part of the SQI Project, a team generated a detailed Software Training Plan for training both managers and engineers about various aspects of developing, managing and acquiring software. The team took the approach of treating the software training program as though it were a

¹ 0-7803-8155-6/04/\$17.00© 2004 IEEE

² IEEEAC paper #1283, Version 8, February 6, 2004

system development task, and went through all the usual phases of development, i.e., requirements, design, “make vs. buy” decisions, implementation, peer review, delivery, and operations.

This paper describes what is involved in a software training program in an engineering and scientific environment. It begins by defining the target customers, required skills sets, and training goals. It then describes the training process utilized at JPL, shows how a consortium of training providers collaborate to provide various types of training, and describes the specific course offerings and implementation approach. It concludes with lessons learned, first in planning a software training approach, and then in actually implementing it with instructors and students.

2. SOFTWARE TRAINING CUSTOMERS

JPL’s employees are classified into 13 job families, and each family has several disciplines and sub-disciplines. While the majority of the JPL Software Community consists of practitioners in the Information Systems and Computer Science (IS&CS) job family, software managers are categorized as either Line Management or Program/Project Management. Also, personnel who are categorized as Engineering and Technical would still be considered part of the Software Community provided that at least 50% of their work is software-intensive. Given this range of categories, the Software Community at JPL consists of approximately 1200 to 1300 people.

The primary training customers are members of JPL’s Software Community, with an initial focus on mission-critical software for spacecraft, instruments, and associated ground systems. This includes project element managers (PEMs), software line managers, cognizant engineers (Cog Es), software systems engineers, software quality assurance (SQA) engineers, mission assurance managers (MAMs), and software practitioners.

Other training customers are managers in JPL program and project offices whose purview is broader than software, but whose scope encompasses it as well. Usually these managers have come from a hardware background, and could benefit from some exposure to the fundamental concepts associated with software management and planning. Hence, other customers include program managers, project managers, systems engineers, and others with whom software personnel interact regularly, and *anyone whose decisions impact the way software is developed at or acquired by JPL*. Lastly, it includes members of the Software Quality Improvement (SQI) Project itself, and selected members of the Acquisition Division involved with acquiring software.

3. SOFTWARE TRAINING REQUIREMENTS

Training requirements were gathered from both external and

internal sources, as indicated below.

External Sources

- NASA directives, guidelines, standards, and lessons learned
- Industry benchmarks (other institutions, IEEE Computer Society Certified Software Development Professional (CSDP), ACM Certified Computing Professional (CCP))
- Industry standards (IEEE, ISO, CMMI)
- Academia (University Software / IT curriculum)

Internal Sources

- JPL studies, reports and lessons learned
- JPL’s Training Advisory Group for the Engineering and Science Directorate
- JPL Flight Projects
- JPL IT and Software Community.

During the requirements collection phase, the team interviewed dozens of managers and software practitioners at all levels of the organization, and asked them what they would like the people in each position of a project to know.

Skill Categories and Associated Skills

Based on the interview results, the team identified and categorized the specific skills needed to successfully manage and develop software. The skill categories could be further characterized as technical or engineering skills, and as managerial or “soft” skills. See Table 1 for a list of these skill categories and associated skills.

The technical skill categories of software management, software engineering, systems engineering, and hardware engineering were to be expected. Finding that the areas of software management that were the most important included software cost estimation, software project planning, and software project monitor and control was also no surprise. However, an equally important finding was the strong emphasis on managerial and “soft” skills, such as vision and leadership, problem solving and decision making, dealing with people, communicating and reporting. While these “soft” skills are not typically associated with your average “software nerd” or “IT geek”, they were deemed critical for the successful and timely management and implementation of software-intensive systems.

Competency Levels

An attempt was made to reach a consensus on the desired level of competence or depth in each of the specific skills. Four competency levels were defined as follows.

- Cursory (C) – Understands basic concepts and terminology
- Working (W) – Understands details, routine

- applications
- Proficient (P) – Solves routine problems
- Expert (E) – Solves complex and unusual problems, consults for others

Skills Matrices

The team then developed a number of skills matrices identifying the specific skills and competency levels for the roles of Project Manager, Project Element Manager, Cognizant Engineer, Software Engineer, Software Architect, SQA Engineer, etc. In addition to identifying the desired skills for each type of role, the matrices also serve as a way to focus the training needed to achieve these levels. In fact, a table of expected training was also developed for each role.

Further analysis of the requirements led the team to determine that software training needs fall into the following five categories:

1. Software management
2. Software engineering
3. Software technology, tools and methodologies
4. Software process improvement
5. Managerial or “soft” skills.

4. SOFTWARE TRAINING GOALS

“One of the specific qualities of knowledge is that it makes itself obsolete very fast. Skills last unchanged for centuries, knowledge changes every few years. If you don’t renew your knowledge often and thoroughly, you become obsolete and fall behind.” Peter F. Drucker, September 2000

The SQI Project wants to ensure that JPL personnel involved in the management, development and acquisition of software systems are adequately trained and possess the knowledge necessary to do their jobs. As such, the project has a number of specific training goals that it hopes to achieve, in collaboration with the other JPL training providers that are part of the JPL Software Training Consortium.

1. Ensure that JPL Project Managers, Project Element Managers, Software Managers and Cognizant Engineers have a good understanding of
 - Software management concepts
 - Software cost estimation
 - Software project planning
 - Software project monitor and control
 - Software risk management.
2. Ensure that JPL software practitioners have a good understanding of
 - Software engineering best practices,
 - Institutional software development process and related processes and procedures,
 - SQI templates and handbooks,

- SQI Software Tool Service (STS).

3. Ensure that Software Quality Assurance (SQA) Engineers and Mission Assurance Managers (MAMs) have a good understanding of
 - Software engineering best practices,
 - Institutional software development process and related processes and procedures, and
 - Software assurance disciplines.
4. Ensure that all SQI Project personnel understand
 - Process improvement,
 - SEI’s Capability Maturity Model Integrated (CMMI), and
 - Organizational change management (OCM).
5. Ensure that the Software Community receives the mandatory IT training specified by NASA and JPL, e.g., annual IT Security training.
6. Ensure that JPL, as an institution, has a training capability that complies with the goals of the CMMI Level 3 Organizational Training (OT) Process Area:
 - A training capability that supports the organization’s management and technical roles is established and maintained.
 - Training necessary for individuals to perform their roles effectively is provided.
 - The training process is institutionalized as a defined process.

To assist in achieving these goals, the Software Training Program is assessed periodically in accordance with the CMMI Organizational Training Process Area by both internal and external auditors, and training metrics for each type of training are collected and reported.

5. SOFTWARE TRAINING PROCESS

The JPL Training Process includes planning and preparation activities, as well as implementation activities. These activities range from requirements collection to conducting courses, and are defined below.

Training Planning and Preparation Activities

1. Collect and analyze software training requirements at least annually.
2. Analyze the gap between training requirements and the current course offerings.
3. Establish the desired software curriculum.
4. Identify new software courses needed and review the purview of training providers.
5. Identify potential internal instructors and/or external providers.
6. Conduct course content development and/or negotiate course content with external providers.

7. Generate periodic updates to existing course content to correct errors or to reflect new practices.
8. Internally review new or modified software training modules.
9. Store instructor and student training materials in a CM-controlled area.

Training Implementation Activities

10. Schedule training rooms, instructors and course offerings.
11. Publish and publicize software training offerings.
12. Register students and maintain training history.
13. Prepare or obtain copies of training materials as needed.
14. Conduct course offerings and gather course evaluations and metrics.
15. Analyze course evaluations and feedback and identify possible updates.
16. Follow-up a representative sample of course attendees to measure training impact.
17. Report training metrics and results at monthly and quarterly management reviews.

Course Content Reviews

In order to ensure consistent quality for each module and to monitor the level of course content, peer reviews or walkthroughs are conducted for each new course under development. Attendees at the peer review include the developer/presenter, the SQI Training Coordinator, the SQI Deployment PEM, and selected reviewers familiar with the topic under review.

6. SOFTWARE TRAINING PROVIDERS

Various organizations and vendors offer technical training at JPL. JPL has an Education and Training Consortium comprised of internal organizations that offer education and training to JPL employees, in support of the technical and business priorities of the Laboratory. Each Consortium member offers training relevant to its own subject area. Four of these training providers offer software training at JPL, and together they form a subgroup now called the Software Training Consortium.

1. HR Professional Development Technical Training Group
2. SQI Project Deployment Element (internally and through other vendors)
3. IT Workforce Enrichment Element of the Center for Space Mission Information and Software Systems (CSMISS) (through the USC Center for Software Engineering and other vendors)
4. IT Education and Training (ITET) Group (internally and through other vendors)

The “wedge” in Figure 2 shows the relationship of these training providers and indicates that the training offered by each of these entities ranges from general role-based training to very specialized training on specific COTS software tools. In addition, courses are offered at JPL and other NASA Centers via the NASA Engineering Training (NET) Program.

7. SOFTWARE TRAINING COURSES

Software training is offered in the following four categories:

- Software management
- Software engineering
- Software technology, tools and methodologies
- Software process improvement

The courses and topics in each category are discussed in more detail below.

Software Management Courses

Currently two software management courses are offered to Project Managers (PMs) and Project Element Managers (PEMs) to give them a general overview of software project planning, and then more details on software project monitor and control. Both courses are offered quarterly and are called Software Management and Planning (SMP), and Quantitative Software Management (QSM).

Software Management and Planning

The purpose of the two-day Software Management and Planning (SMP) course is to provide experienced JPL Project Managers and PEMs with relevant information about software issues that may affect the success of their projects. The goals of this course are to provide an increased understanding of the software issues relative to the management of projects, and to give managers the ability to more effectively manage the software in their projects. Topics covered in the course include:

- Overview: Software Trends and Lessons Learned
- Software Project Planning
- Software Life-Cycles
- NASA and JPL Standards
- Software Project Monitor and Control
- Software Metrics
- Staffing Considerations
- Software Acquisition
- Systems Engineering Considerations
- COTS Software and Reuse of Software
- Software Cost Estimation
- Software Requirements Management
- Software Testing
- Software Quality
- Software Configuration Management (CM)
- Software Development Environments
- Software Management Summary

Quantitative Software Management

The purpose of the two-day Quantitative Software Management course is to train PEMs and Cognizant Engineers in how to generate preliminary and detailed cost estimates and schedules for the entire software development life-cycle. Topics covered include:

- Software estimating methods and models
- Software development estimation steps
- Software development metrics
- Software cost drivers
- Productivity
- Common errors in software estimation
- Rules of thumb
- Software productivity databases
- Software risk reduction and mitigation.

Software Engineering Courses

Currently three software engineering courses are offered to Cognizant Engineers (Cog Es) and Software Engineers, including Software Product Engineering, Software Peer Reviews, and Software Testing.

Software Product Engineering

The purpose of the one-day Software Product Engineering course is to provide more detailed instruction to JPL Cog Es and software developers about various aspects of the software life-cycle, as well as the methodologies and tools which facilitate the development process. Topics covered in the course include:

- Software Process Overview
- Software Methodologies and Tools
- Software Requirements Definition and Analysis
- Software Architecture
- Data Engineering, Data Modeling, and Data Management
- *Software Design (future)*
- Software Testing and V&V
- Peer Reviews
- Software Cost Estimation
- Managing and Coordinating Interfaces

Software Peer Reviews

The purpose of the one-day Software Peer Reviews course is to train PEMs and Cog Es in how to conduct various types of peer reviews of software work products ranging from desk checks and walkthroughs to formal inspections. It covers activities such as planning and conducting the peer review itself through creating both detailed and summary reports and metrics collection. Checklists are provided for various types of inspections including system requirements, subsystem functional requirements, software requirements, architectural design, detailed design, code (language

specific), test plans and test procedures.

Software Testing

The purpose of the two-day Software Testing course is to provide software engineers and SQA engineers with a thorough understanding of testing approaches, techniques and tools. Topics covered include test planning and management, organizing and developing test cases and reports, unit testing, integration testing, acceptance testing, system testing, and maintenance and regression testing.

System Software Reliability

The purpose of the three-day System Software Reliability course is to train software engineers, SQA engineers, systems engineers, or other individuals responsible for measuring, analyzing, designing, automating, implementing or ensuring software reliability. Topics covered in the course include:

- History of Software Reliability
- Software Reliability Defined
- Software Life-Cycle
- Factors That Impact Software Reliability
- Overview of Software Reliability Models
- Data Required for Models
- Software Reliability Prediction Models
- Software Reliability Estimation Models
- Software Reliability Metrics
- Software Fault Trees
- Software Failure Mode and Effects Analysis
- System Reliability, Software Redundancy
- Improving Software Reliability
- Managing Software Reliability

System Requirements and Management

The purpose of the three-day System Requirements and Management course is to teach systems engineers, software engineers, and managers about the different aspects of requirements management over the life of the project, particularly how to define the scope of their project, which is essential to writing good requirements. They learn how to recognize poor requirements, write good requirements, and use rationale to clarify requirements. Topics covered in the course include:

- Define Project Scope
 - Define needs, goals, and objectives
 - Identify stakeholders
 - Develop operational concepts
 - Define interfaces
- Write Good Requirements
 - Understand what makes a good requirement
 - Understand different types of requirements

- Expand a standard specification
- Create checklists for requirements
- Write rationale for requirements
- Define verification methods
- Develop traceability
- Manage Requirements
 - Understand what needs to be managed
 - Control change
 - Define and control priorities
 - Allocate requirements
 - Identify and control risks
 - Ensure the proper reviews
 - Collect meaningful metrics

Software Process Improvement Courses

The software process improvement training is primarily focused on the SEI's Capability Maturity Model Integration (CMMI). Four courses are offered including Overview of CMMI, Introduction to CMMI, Intermediate CMMI, and Mastering Process Improvement. While these courses are primarily meant for the SQI Project personnel, process engineers, system engineers and any others involved in process improvement, senior managers and other managers also take the overview course.

Software Technology and Tools Training

Short seminars and tutorials sponsored by the SQI Software Tools Service (STS) are offered by various vendors on their commercial-off-the-shelf (COTS) tools for use in the software development process, including CASE tools, operating systems, languages, debugging tools, and test tools, etc.

8. MANAGERIAL TRAINING COURSES

Training in managerial and "soft" skills is offered by the JPL Professional Development Section. Role-based managerial training is provided through 5-day courses such as the following:

- The JPL Project Manager
- The JPL Project Element Manager
- The JPL Task Manager
- Contract Technical Management
- The JPL Group Supervisor
- The JPL Cognizant Engineer

These role-based courses cover a broad spectrum of topics including roles and responsibilities throughout the project life-cycle, JPL as an institution and how to work in the JPL environment, JPL's relationship to NASA, JPL's process structure and how these processes apply to projects, the NASA/JPL project life-cycle, how to plan and manage a

project element or task, mitigating risks, estimating costs, developing budgets, description of the resources available to help and how to use them, etc. The first three of the courses above also contains a short module on software development and acquisition. The JPL Cognizant Engineer course contains eight modules covering software development in various application domains ranging from flight software to ground software.

Additional managerial training is provided through focused courses such as:

- Cost Planning, Scheduling, Estimating and Performance Management
- How to Balance Priorities
- Time Management and Organization Skills for Managers and Supervisors
- The Engineer in Transition to Management

Training in "soft" skills is provided through courses such as:

- Leadership and Emotional Intelligence
- How to Handle People with Tact and Skill
- How to Handle Difficult People
- Effective Communication for Managers and Supervisors
- Negotiation Skills for Managers and Supervisors
- Skills For Managing Conflict and Reaching Resolution
- Presentation Skills
- Creating the Coaching Environment
- Succeed Over Stress

9. TRAINING IMPLEMENTATION

At the beginning of each fiscal year, the team performs steps #1 through #9 of the software training process (See Section 5), and schedules courses for the year. Students register for courses on-line using an automated registration and tracking tool called Registrar.

The role-based training is typically offered three times per year, as is the software management training, which is scheduled to follow closely afterwards. Recently, these software management courses have been offered more frequently to specific sections that are participating in a more intensive software improvement effort. The software engineering training is also offered three times per year, and is also scheduled to follow closely after the role-based training for Cognizant Engineers.

Over the past three years, approximately 700 students have been trained in various aspects of software management, software engineering, and software process improvement. See Figure 1 for a graph of the cumulative training provided thus far. Of course, some employees have taken more than one course so this total does not indicate that the overall

target audience has been reached yet.

In addition to tracking course offerings and attendance, JPL also has students complete evaluation forms for each course module, and for the course overall, in order to monitor the quality of each module, gauge various instructor's presentation styles, and discern what changes, if any, should be made. Hence, the student evaluations assist in the continuous process improvement of the course offerings.

10. JPL LESSONS LEARNED

The JPL SQI Deployment Element has collected a number of "lessons learned" from planning and delivering a software training program in an engineering and scientific environment. The lessons fall into two categories: a.) lessons about developing a training approach and generating a Software Training Plan, and b.) lessons about implementing and carrying out the actual training itself. *Software Training Approach and Training Plan*

1. Allocate sufficient time to lay the groundwork for developing the training approach and actually writing the training plan. The time allocated should be on the order of months, not weeks. Allocate time to:
 - Gather requirements
 - Document what courses are already being offered
 - Review the CMMI Organizational Training process area
 - Coordinate with other training providers, including JPL, NASA and third party vendors.
 2. Promote communication among the various training providers and clarify the roles each will play, ranging from role-based to overview to detailed tools (ala "the wedge" in Figure 2).
 3. Develop skills matrices for various roles to help clarify what training is needed overall and to what depth, i.e., what courses to offer. Also, it helps supervisors know what courses to promote to their group members during performance evaluations.
 4. Adopt a few basic categories and similar terminology for competency levels as HR Professional Development to help in curriculum discussions and aid in determining content level.
 5. Be sure to distinguish between training needs that are the responsibility of the training organization and those that are the responsibility of the projects themselves
- Software Training Implementation*
1. Allocate sufficient lead time for course content development.
 - It can take several months for each module since instructors are not full-time. They're experts so they're out plying their trade.
 2. Establish presentation templates and enforce strict configuration management on course content.
 - Conduct a peer review or "dry run" of course content.
 - Some modifications are needed occasionally to ensure that content reflects latest standards, processes, and trends.
 3. Have students complete evaluation forms for each course module and for the course overall. This provides valuable information needed for continuous process improvement.
 4. Allocate sufficient time for logistics coordination and hire a part-time administrator to do it. This person needs to:
 - Schedule classrooms,
 - Coordinate instructor availability,
 - Send invitations for nominations to managers of software-intensive sections,
 - Enroll students in "By Invitation Only" courses,
 - Send e-mail notices and reminders, etc.
 5. Develop a process sheet for the myriad of tasks to complete before each course offering to keep items from falling through the cracks and to ensure necessary lead times.
 6. Begin work on logistics at least six weeks before a class is to be offered.
 7. Offer courses by "Invitation Only," upon recommendation of managers, rather than by "Open Enrollment". This provides not only more attendees, but also more appropriate roles.
 8. Overbook registration by at least 15% in order to guarantee a full class since invariably, at the last minute, someone fails to show up or cancel.
 9. Ensure that presenters have the appropriate presentation skills, in addition to their domain expertise, and that they provide "real world" cases to turn their concepts into concrete examples.
 10. Be flexible and open to change the sequence of modules in order to accommodate instructor availability, and

even instructors for various modules, over time, due to changing assignments, time pressures and personal interests.

11. CONCLUSIONS

In order to maintain a vital software community and ensure a trained software workforce, companies must provide for career growth and enhancement of technical skills for software professionals. Given the competition for software professionals, sometimes “growing your own” is the only way to ensure certain skills are available when you need them. Dr. Beverly Kaye, CEO of Career Systems International and co-author of *Love ‘Em or Lose ‘Em*, has a great response for any manager who complains about the cost of training someone who then leaves: “Yes, but think of the cost of NOT training them and then having them stay!” Hence, training needs to be viewed as a “cost of doing business”. Providing adequate and timely training includes identifying courses, developing curriculum, finding trainers, and conducting courses or procuring training elements from vendors. Each of these activities takes time, money, and commitment. A side benefit is that training promotes retention of employees, since most people enjoy learning something new and appreciate being given the opportunity to stay current technically.

Software training needs to be geared towards specific competency levels, i.e., everyone does not need to become an expert in every aspect of software development and software management. A pathway needs to be provided for software engineers to make the transition to Software Cognizant Engineer, and perhaps, eventually to Software Manager. The skills set for software management is entirely different than that for software engineering. Understanding software cost estimation and software planning is very important, but even as much so, is software project monitor and control. Managers must ensure that the system is actually delivered and is within the allocated schedule and budget.

Lastly, software training needs to include not only technical skills, but also managerial and “soft” skills. It is important to realize that no amount of processes, procedures or training can make up for lack of communication or teamwork!

12. ACKNOWLEDGEMENTS

Many people have contributed to the success of JPL’s software training program and deserve recognition.

- Brian Vickers – SQI Software Training Coordinator
- Michelle Medina – SQI Software Training Logistics Support
- K. Virginia Choate – IT Education and Training Lead
- Brindley McGowan – Group Supervisor, Technical Curriculum Development Group, HR Professional

Development

- Sandra Dennis – Training Systems Operations Group, HR Professional Development
- Ray Kile and Dave Rolley – CMMI Auditors from CSM, evaluation of JPL’s CMMI Organizational Training (OT) Process Area (PA).

All of the instructors have spent time developing the actual course content and instructing classes: Robert Barry, Erich Corduan, Dan Crichton, William Decker, David Eisenman, Dan Erickson, Alan Ferdman, John Hackney, Dave Hermsen, Jairus Hihn, Terry Himes, Frank McGarry, Ron Morillo, George Rinker, Nicolas Rouquette, Kimberly Simpson, Tuyet-Lan Tran, and former instructors Milton Lavin, Burt Sigal and Jody Steinbacher. Nor should we forget over 700 students who have attended our courses and provided valuable feedback.

The work described in this paper was performed at the Jet Propulsion Laboratory, California Institute of Technology under a contract with the National Aeronautics and Space Administration (NASA). Reference herein to any specific commercial product, process or service by trade name, trademark, manufacturer, or otherwise, does not constitute or imply its endorsement by the United States Government, NASA or the Jet Propulsion Laboratory, California Institute of Technology.

REFERENCES

- [1] U.S. Office of Personnel Management, Office of Workforce Relations, *A Guide to Strategically Planning Training and Measuring Results*, OWR-35, July 2000
- [2] Watts S. Humphrey, *A Discipline for Software Engineering*, Addison-Wesley, New York, 1995
- [3] Mary Beth Chrissis, Mike Konrad, Sandy Shrum, *CMMI: Guidelines for Process Integration and Product Improvement*, Addison-Wesley, San Francisco, 2003
- [4] *Capability Maturity Model Integration, Version 1.1*, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, 2002
- [5] *People Capability Maturity Model, Version 2.0*, Carnegie Mellon University, Software Engineering Institute, Pittsburgh, PA, 2001
- [6] Donald L. Kirkpatrick, *Evaluating Training Programs: The Four Levels*, Berrett-Koehler Publishers, Inc., 2nd edition, July 1998
- [7] Beverly Kaye and Sharon Jordan-Evans, *Love ‘Em or Lose ‘Em: Getting Good People to Stay*, Berrett-Koehler Publishers, Inc., San Francisco, 1999

[8] Beverly Kaye and Sharon Jordan-Evans, *Love It, Don't Leave It: 26 Ways to Get What You Want At Work*, Berrett-Koehler Publishers, Inc., San Francisco, 2003

[9] Barry W. Boehm, et al, *Software Cost Estimation with COCOMO II*, Prentice-Hall PTR, Prentice-hall Inc., Upper Saddle River, New Jersey, 2000

[10] *NASA Program and Project Management Processes and Requirements*, NASA Procedures and Guidelines, NPG 7120.5A, 1998

[11] *NASA Software Management, Engineering, and Assurance*, NASA Procedures and Guidelines, NPG 2820

[12] *IEEE Computer Society Certified Software Development Professional Certification Requirements*
<<http://www.computer.org/certification/>>

[13] *IEEE Computer Society Distance Learning Center*
<<http://www.computer.org/distancelearning/>>

[14] *ACM Professional Development Centre (PDC)*
<<http://pd.acm.org/>>

[15] *Institute for Certification of Computing Professionals*
<<http://www.iccp.org/intro.html>>

BIOGRAPHY



P. A. "Trisha" Jansma is the Project Element Manager (PEM) for the Deployment Element of the Software Quality Improvement Project at the Jet Propulsion Laboratory (JPL) in Pasadena, CA. With over 30 years at JPL in both line and task management positions, she has a broad background in systems and software engineering and information technology, in engineering and scientific environments. Jansma has extensive experience in the management, design, development and delivery of cost-effective, software-intensive systems. She has experience in all facets of project life-cycle development, from initial feasibility analysis, proposal development and conceptual design through documentation, implementation, user training, enhancement and operations. Jansma has a B.A. in Mathematics from Point Loma Nazarene University, an M.S. in Computer Science from the University of Southern California, and an Executive M.B.A. from the Peter F. Drucker Graduate School of Management at Claremont Graduate University. She also holds a California Community College Teaching Credential and a California Secondary Teaching Credential, and has taught Systems and Software Engineering courses at the graduate level.

Table 1. Skill Categories and Skills

Engineering or Technical Skills	Managerial or “Soft” Skills
<ul style="list-style-type: none"> • Software Management <ul style="list-style-type: none"> – Software Cost Estimation – Software Risk Management – Software Project Planning – Software Development Environments – Software Project Monitor and Control – Software Measurement and Metrics – Software Quality Assurance – Software Configuration Management 	<ul style="list-style-type: none"> • Project Planning and Tracking <ul style="list-style-type: none"> – Task Planning and Task Allocation – Schedule Development and Tracking – Budget Development and Tracking • Problem Solving and Decision Making <ul style="list-style-type: none"> – Problem Identification, Solution, Escalation – Timely Decisions, Follow Through
<ul style="list-style-type: none"> • Software Engineering <ul style="list-style-type: none"> – Software Architecture – Software Design – Software Implementation – Software Reliability and Safety – Software Verification and Validation – Methodologies, Tools, Processes – Software Technology Awareness – Application Domain-Specific Knowledge 	<ul style="list-style-type: none"> • Vision and Leadership <ul style="list-style-type: none"> – “Big Hat”, “Big Picture” Approach – Clear Picture of Problem – Ownership of Problem
<ul style="list-style-type: none"> • Systems Engineering <ul style="list-style-type: none"> – Requirements Definition and Analysis – Tradeoffs, Tailoring, Prioritizing – System Architecture – Testing Approaches – Processes, Procedures, ISO 	<ul style="list-style-type: none"> • Dealing with People <ul style="list-style-type: none"> – Staffing, Team Selection – Team Building – Conflict Resolution – Delegating – Negotiating – Challenging, Inspiring, Motivating
<ul style="list-style-type: none"> • Hardware Engineering <ul style="list-style-type: none"> – Hardware Architecture and Design – Hardware Safety and Handling – Firmware – Hardware Test and Validation – Hardware Technology Awareness 	<ul style="list-style-type: none"> • Communicating and Reporting <ul style="list-style-type: none"> – Presentations, Reviews, Reports – Customer Focus and Awareness – Sponsor Interface – Communication with Team and Management – Meeting Management

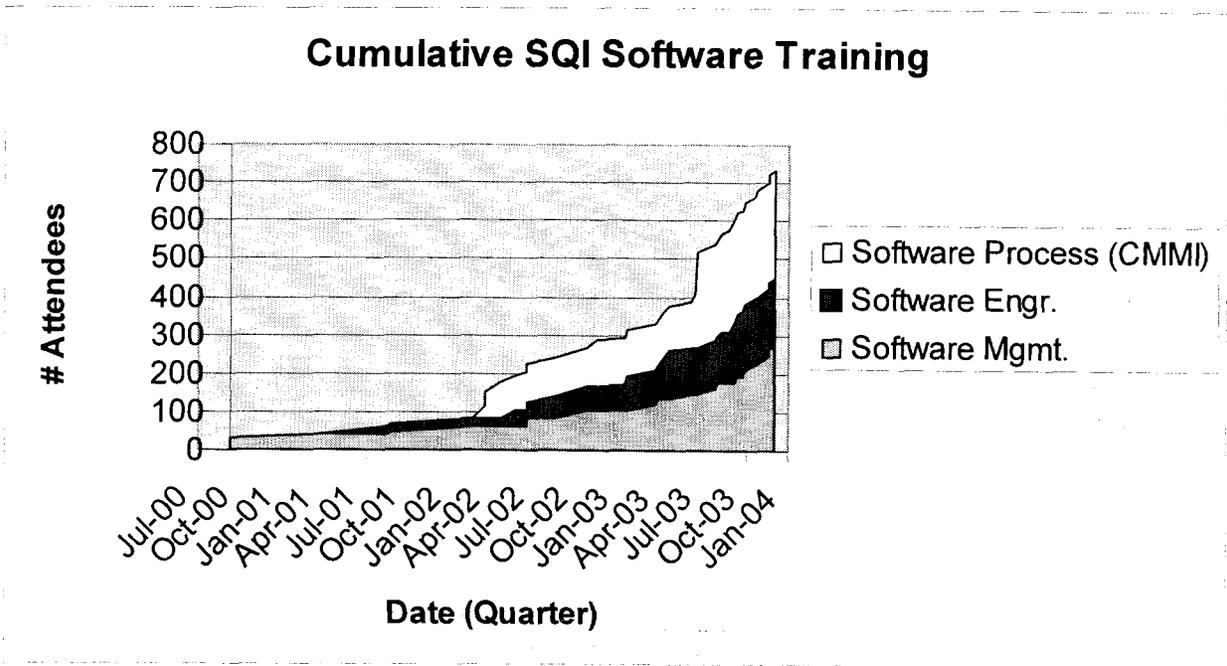


Figure 1 Cumulative SQA Software Training

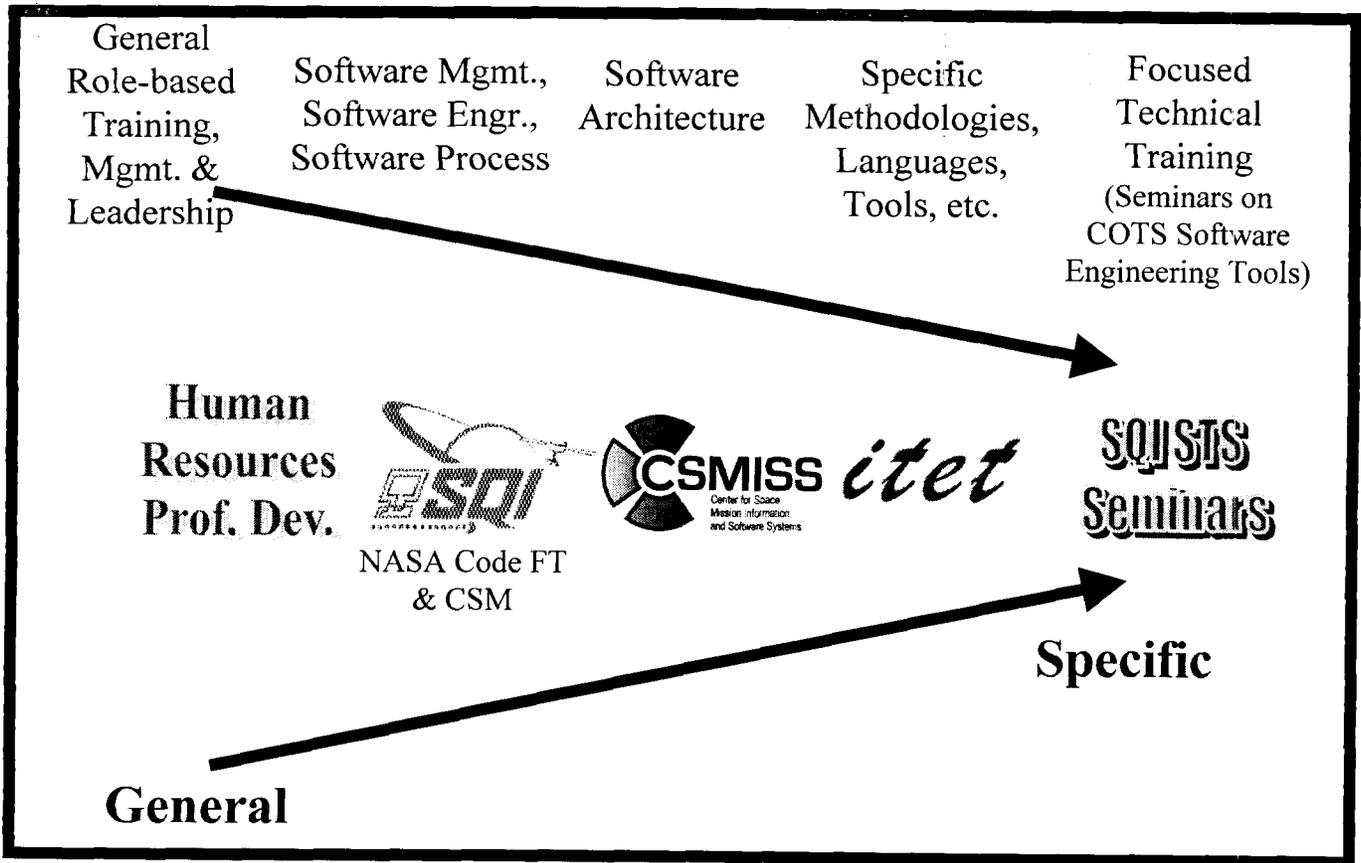


Figure 2. JPL Software Training Consortium -- "The Wedge"