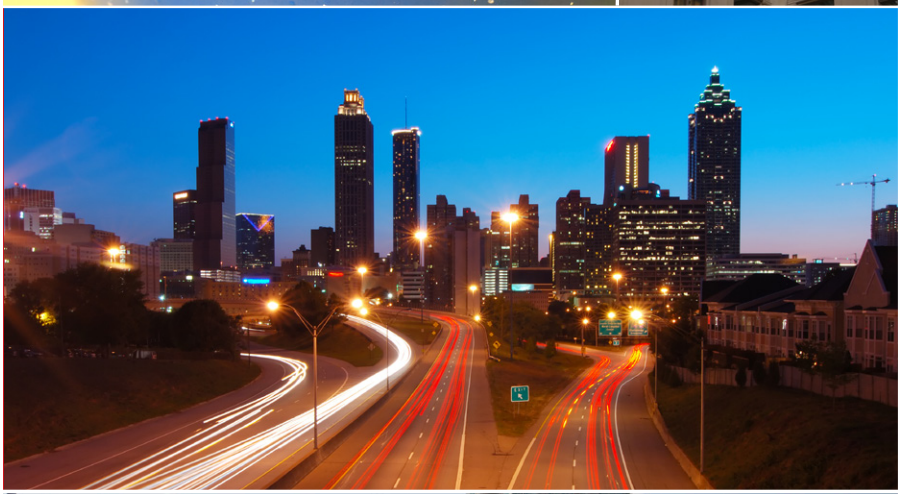
TSP SYMPOSIUM 2011
A Dedication to Excellence

6th Annual Software Engineering Institute (SEI)
Team Software Process (TSP) Symposium

September 19-22, 2011 • Georgia Tech Hotel and Conference Center • Atlanta, Georgia

Software Engineering Institute | Carnegie Mellon



www.sei.cmu.edu/tspsymposium

TSP SYMPOSIUM 2011
A Dedication to Excellence

6th Annual Software Engineering Institute (SEI)
Team Software Process (TSP) Symposium

Table of Contents

Abstracts

Proceedings

Welcome

Welcome to Atlanta for the sixth annual Team Software Process (TSP) Symposium. Last year, we had the pleasure of welcoming delegates to the Carnegie Mellon Software Engineering Institute's (SEI) headquarters city of Pittsburgh. This year, I hope that you feel a continued sense of hospitality that is the trademark of this southern city.

The TSP Symposium 2011 theme is *A Dedication to Excellence*, in honor of the late Watts Humphrey who was commonly referred to as the "Father of Software Quality." Among his many accomplishments—including founding the Software Process Program at the SEI and receiving the National Medal of Technology—Humphrey played a significant role in the development of the Team Software Process. Humphrey, 83, died October 28, 2010, at his home in Sarasota, Florida.

So, it is most fitting that the focus of this year's symposium comes from Humphrey's book, *TSP: Leading a Development Team.* The phrase *A Dedication to Excellence* not only describes a self-directed TSP team, it also describes the man who helped create it. It is the dedication to excellence of those who advocate and use the TSP in their daily professional lives that has transformed TSP into what it is today and what it has yet to become.

It is hard to believe TSP has been around for more than a decade. While much has changed since Watts wrote the first technical report on TSP back in 2000, the TSP's abilities have remained constant. Among its benefits, TSP provides adaptive planning, process, and quality frameworks which allows teams to incorporate the appropriate technical techniques for the given environment, while still meeting cost, schedule, and quality commitments.

On behalf of the TSP Symposium 2011 program committee, we have been thoroughly impressed with the quality of this year's technical sessions. As you will hear from your colleagues from around the globe, TSP is not only one of the top-rated software engineering methods for small, medium, and large systems, it has also proven beneficial to other disciplines as well, such as systems engineering, testers, process improvement teams, and nuclear engineers, just to name a few. While there is variation in the engineering practices used by successful projects, they are all disciplined in producing excellent products and services to their customers and users.

I hope you find the TSP Symposium 2011 to be a valuable learning and networking opportunity and enjoy the coming together of TSP practitioners from around the world. To help facilitate the open exchange of ideas, we are excited to implement "Best Practices," a brand-new way to share new ideas, concepts, practices, or takeaways you've discovered while in session or in conversation with your fellow attendees. Read more about this on page 2.

My hope is that you take these next few days at the TSP Symposium 2011 to share common experiences and absorb new lessons learned in the adoption of TSP, as well as the benefits we have experienced in applying TSP in both software and other knowledge-worker environments.

Timothy A. Chick
Software Engineering Institute, TSP Symposium 2011 Technical Chair
Senior Member of Technical Staff, SEI Software Engineering Process Management Program

Attendee Information

General Information
All TSP Symposium 2011 sessions, continental breakfast, and breaks will be held on the second floor of the Georgia Tech Hotel and Conference Center. Lunch will be served in the dining room on the first floor. Please check individual listings in this program for specific locations and times for all presentations and social functions. Any changes in the program schedule will be posted or announced during the general session.

Registration Hours
The registration desk is located in the Grand Ballroom foyer area. Hours are

Monday, September 19	8:00 a.m. to 3:30 p.m.
Tuesday, September 20	7:30 a.m. to 3:30 p.m.
Wednesday, September 21	7:30 a.m. to 3:30 p.m.
Thursday, September 22	7:30 a.m. to 3:30 p.m.

Smoking
The Georgia Tech Hotel and Conference Center is a non-smoking facility. Smoking is permitted only in designated areas.

Cellular Telephones and Beepers
As a courtesy to the speakers and fellow attendees, we request that all devices be switched to silent or vibrate mode in session rooms during all presentations.

Message Board
As a service to the TSP Symposium 2011 attendees, a message board will be located at the registration desk in the Grand Ballroom foyer area. Conference staff will monitor the message board during open registration hours; unclaimed messages will be removed daily.

Birds-of-a-Feather Sessions

Tuesday, September 20	5:00 p.m. to 6:00 p.m.
Wednesday, September 21	5:00 p.m. to 6:00 p.m.

The TSP Symposium 2011 provides a forum for people to meet informally in a relaxed setting. Any conference participant can create a one-hour Birds-of-a-Feather (BoF) session. To do so, add your name and topic to the sign-up sheets at the registration desk. There are two rooms available for this purpose and rooms will be assigned to the sessions with the most interest. A minimum of 15 people is required for room assignment.

Recommended Attire
Recommended attire for the TSP Symposium 2011 activities is business casual. Because meeting rooms can often be cold, we recommend you bring a light sweater or jacket with you to ensure your comfort.

Personal Valuables and Lost and Found
Do not leave any valuables such as laptop computers or purses unattended as the SEI is not responsible for any lost items. Additionally, there is no place to secure valuables in the meeting rooms so we ask you to please be sure to take them with you when you leave.

During the TSP Symposium 2011, conference staff will maintain a lost and found at the registration desk in the Grand Ballroom foyer. Once the TSP Symposium 2011 concludes, any unclaimed items will be turned into the Georgia Tech Hotel and Conference Center's lost and found.

Best Practices, Tips, and Tricks—New for 2011

With your registration materials, you'll find several "Best Practices" forms. Throughout the week, these forms will let you share any new ideas, concepts, practices, or takeaways you've discovered while in session or in conversation with your fellow attendees.

Simply jot down the idea you'd like to share and drop off the yellow copy at the registration desk. You may also email your Best Practices to tsp-symposium-info@sei.cmu.edu, if you prefer. We will post the ideas on a message board in the conference area. They may spark conversation with your fellow attendees, topic ideas for the Birds of a Feather sessions, or questions for speakers.

There is no limit on the number of ideas you can share. The conference staff will have plenty more forms available in the session rooms and the registration desk. After the TSP Symposium 2011 concludes, we will compile all of the submitted tips into one document and share it with all attendees.

Hotel Information

Hotel Contact Information
Georgia Tech Hotel and Conference Center
800 Spring St. NW, Atlanta, GA 30308
Phone: 404-347-9440
Fax: 404-347-9088

Business Center
Grand Ballroom, Second Floor, 7:00 a.m. to 6:00 p.m.

Stay connected and productive away from your office at the Georgia Tech Hotel and Conference Center's full-service business center. Amenities include computers with high-speed Internet access, as well as print, copy, and fax capabilities. The business center can also handle your shipping and delivery requirements.

Additionally, the Georgia Tech Hotel and Conference lobby kiosk offers complimentary Internet access and printer for last-minute airline boarding passes.

First Aid and Emergencies
In case of a medical or other personal emergency during conference hours, please proceed to the Business Center (Grand Ballroom, Second Floor) and the attendant will call Security. Security will call 911 and activate the necessary personnel (police or ambulance) to come. After hours, contact the front desk or Security directly for assistance.

In case of a weather emergency in Atlanta (tornado, hurricane, or severe storm) TSP Symposium 2011 attendees should seek shelter in the Grand Ballroom, second floor. In the event of a fire or other evacuation, simply exit down the nearest stairwell and out of the building. We will assemble in the parking lot at the corner of 5th St. NW and Spring St. NW.

Nearest Medical Center
Emory University Hospital Emergency Department
1364 Clifton Rd. Atlanta, GA 30320
Phone: 404-712-7100
Open 24 hours a day, 7 days a week.

For non-emergencies
(including dental and optometrist requirements)

Emory University Hospital Midtown
550 Peachtree St. NE, Atlanta, GA 30308
For information on doctors and the conditions they treat, call the Emory University Hospital Midtown Physician Referral Line at 404-778-2000.

If you have medical questions, or need to make an appointment, you can speak to a nurse by calling 404-778-7777 or 1-800-75EMORY (1-800-753-6679).

Emergency Considerations
To help ensure your safety while you are attending the TSP Symposium 2011, we recommend the following:

- If your emergency contact information has changed since you registered for the TSP Symposium 2011, please complete the emergency contact card in your registration package and drop it off at the registration desk to update your records.
- If you prefer, complete the emergency contact card and keep it on your person during your stay.
- Consider programming an "ICE" number into your cell phone. This is your preferred contact "in case of emergency" and assists emergency personnel in contacting the correct person in your phone book. Simply store the preferred number with a name starting with "ICE."
- Take a moment to acquaint yourself with the emergency exits and contacts within your hotel in case you experience an emergency after hours. In most hotels, this is typically the front desk and/or the security department.
- In the event of an emergency during the TSP Symposium 2011, please follow the directions of TSP staff and hotel personnel.
- Please practice good laptop security. When leaving the meeting room, please take your laptop and any other electronics with you. If you have a laptop locking cable please consider using it in the meeting rooms and in your hotel room.
- When leaving the Georgia Tech Hotel and Conference Center, please make sure to remove your name badge.

Emergency Preparedness Plans
If an emergency should take place during the TSP Symposium 2011, please be aware that the Georgia Tech Hotel and Conference Center has an emergency plan in place. In the event of an emergency, all attendees would be directed by hotel personnel to a staging area away from the particular threat.

In addition, all evacuation plans are on the back of each hotel room door to guide you to the appropriate exit should you be in your room at the time of an evacuation.

Parking
Overnight parking is offered in the Georgia Tech Hotel and Conference Center's covered parking garage for overnight guests. This covers entrance and exit only, no in and out privileges.

Local transportations options can be found by visiting the Metropolitan Atlanta Rapid Transit Authority's website at www.itsmarta.com.

About Atlanta

Atlanta, the host city for the TSP Symposium 2011, is known as a world-class, modern city with a rich, passionate history. The perfect backdrop for four days of engaged learning, Atlanta offers an impressive legacy of leadership, progress, and inspiration. Midtown offers a blend of dining, cultural attractions, and nightlife in an urban setting with the comfort of an increasingly residential area. Known as the heart of the arts in Atlanta, Midtown houses the Atlanta Botanical Garden, Piedmont Park, the Woodruff Arts Center, the Fox Theater, bungalows, skyscrapers, restaurants, and churches.

That's just the beginning. There is plenty to do and see in the Midtown area alone. Here is just a sampling of what there is to see and do in the area near the Georgia Tech Hotel and Conference Center:

- **Visit the Atlanta Botanical Garden**: With more than 30 acres of gardens, forest, wildflower trails, and a 10,000 square-foot Fuqua Orchid Center, the Atlanta Botanical Garden is one of Atlanta's most beautiful attractions. The Canopy Walk, a 40-foot-high suspension bridge creating a treetop walking trail is the garden's latest addition. Visitors will also enjoy the Edible Garden Outdoor Kitchen and soothing sanctuary of the Cascades Garden.
- **Take a tour of the Atlanta Brewing Company**: Atlanta Brewing Company (ABC) tours teach the history and how-to of craft brewing. As Georgia's oldest operating craft brewery, ABC is a Southern favorite. Prior to tours, enjoy a happy hour tasting where you can enjoy the season's newest additions well as tried-and-true brews.
- **Enjoy the luxury and history of the Fox Theatre:** Designed in the late 1920s, this opulent, historic landmark was originally the Yaarab Temple Shrine Mosque. Today, The Fox Theatre presents shows by Broadway in Atlanta and Theater of the Stars, as well as a summer movie series.
- **Drop in at the Margaret Mitchell House & Museum:** The three-story Tudor Revival mansion in Midtown was the home of author Margaret Mitchell, where she wrote the Pulitzer Prize-winning novel, *Gone With the Wind*. The house offers tours, a museum, and shop, as well as a literature series.
- **Enjoy the seasonal September weather at Piedmont Park:** Piedmont Park features lush woods, Lake Clara Meer, picnic spots, skating paths, and many annual events. More than 180 acres, the park is situated between 10th Street and Piedmont Avenue, where it joins the Atlanta Botanical Garden.
- **Enjoy some friendly competition at Ten Pin Alley:** An upscale restaurant and lounge, Ten Pin Alley offers premium billiards and bowling. This sophisticated yet exciting venue serves a gourmet American menu, paired perfectly with an extensive wine, beer, and cocktails list.

For more information on attractions in Midtown and throughout the Atlanta area, visit the website of the Atlanta Convention & Visitors Bureau: www.atlanta.net.

Keynote Speakers

Keynote Speaker: Capers Jones
Capers Jones is currently the president and CEO of Capers Jones & Associates LLC. He is also the founder and former chairman of Software Productivity Research LLC (SPR). He holds the title of Chief Scientist Emeritus at SPR. Jones founded SPR in 1984.

Before founding SPR, Jones was assistant director of Programming Technology for the ITT Corporation at the Programming Technology Center in Stratford, Connecticut. He was also a manager and researcher at IBM in California.

Jones is a well-known author and international public speaker. Some of his books have been translated into six languages. All of his books are translated into Japanese and his newest books are available in Chinese editions as well.

His most recent book was *Software Engineering Best Practices* which was published by McGraw Hill in October 2009. Jones's next book will be an *Encyclopedia of Software Quality,* also for McGraw Hill with Dr. Bill Curtis as a co-author. In addition to his technical books, Jones has also received recognition as an historian after the publication of *The History and Future of Narragansett Bay* in 2006 by Universal Publishers.

Jones and his colleagues have collected historical data from more than 600 corporations and more than 30 government organizations. This historical data is a key resource for judging the effectiveness of software process improvement methods. More than 13,000 projects have been reviewed. This data is also widely cited in software litigation in cases where quality, productivity, and schedules are part of the proceedings. Jones has worked as an expert witness in 15 software-related lawsuits dealing with breach of contract and also with software tax issues.

His research studies include quality estimating, quality measurement, software cost and schedule estimation, software metrics, and risk analysis.

Jones has consulted at more than 150 large corporations and also at a number of government organizations such as NASA, the U.S. Air Force, U.S. Navy, Internal Revenue Service, and the U.S. Courts. He has also worked with several state governments.

Keynote Speaker: Michael Sapenter

Michael D. Sapenter is the vice president, Software Engineering & Integration Division (SEID), for CGI Federal, Technical Program Group. He is currently responsible for System and Software Engineering Services associated with development, deployment, post-production software support, information technology, information assurance, configuration management, quality assurance, testing, interoperability, simulation/training, and training for Command & Control, Target Acquisition, Meteorological, and communication systems.

Sapenter began his career working for Litton Data Systems Division developing software for the U.S. Army's initial Command and Control system. In 1976 he joined Techrizon (formerly TELOS Corporation) as a software programmer working on the Fire Support contract, a contract with the U.S. Army to develop software for various weapons systems. He has held numerous software development positions associated with designing and implementing solutions for Command and Control, Fire and Effects, and operating and communication systems.

Additionally, he participated on a team formed to develop and implement automated test and interoperability tool sets for the validation of the Command and Control, Fire and Effects, Target Acquisition, and Meteorological systems. He chaired the initial working group for the implementation of software engineering methods, technologies, and quality processes. While performing duties as the director of engineering, Stanley—now a wholly-owned subsidiary of CGI Federal, Inc.—achieved the Capability Maturity Model Integration (CMMI) Level 5 rating using the Version 1.1. Most recently, as the vice President of SEID, the division achieved a CMMI Level 3 rating using Version 1.2 of the model.

Before obtaining his current position, Sapenter held numerous managerial positions within Techrizon to include section manager, branch manager of System Production, director of engineering, and deputy program manager of the Fire Support contract.

Sapenter holds a bachelor's degree in mathematics from Prairie View A&M University and a master of business administration degree from Oklahoma City University.

Keynote Speaker: Carl Wyrwa

Carl Wyrwa is director of quality, with a focus on software, at Beckman Coulter, Inc. located in Brea, California. He is responsible for the safety, quality, and compliance of the medical device software integrated into the diagnostic products developed by the company. He is responsible for the implementation of the TSP company-wide.

Wyrwa has been with the company since 1974 where he began as an engineering programmer for corporate data processing initially supporting the corporation's time-sharing system and providing scientific applications software support. He was assigned in 1975 to develop the software for the first fully automated computer controlled blood analyzer which was introduced to the clinical market in 1978. He continued in the area of software development for the blood analyzer products and formed a group of 135 software developers responsible for the development of all medical device software until 1993.

In 1993 he rejoined the corporate data processing group, leading the organization in its efforts to centralize the information technology support for the corporation worldwide. He also introduced the company to the concept of Enterprise Resource Planning (ERP) systems which resulted in the transition to a corporate ERP system worldwide.

Upon the acquisition of new entities developing medical devices in 1997, he assumed a corporate role of standardizing the software development activities and the software quality assurance oversight activities company-wide.

He has been an active member of the AdvaMed (Advanced Medical Technology Association) software working group since 1986, which has included planning AdvaMed software conferences and speaking at the conferences. He has also been active in AAMI (Association for the Advancement of Medical Instrumentation) Technical Information Report writing activities, working for several years with FDA and industry on TIR36:2007, "Validation of Software for Regulated Processes."

Wyrwa is a member of the advisory committee for the medical product development certificate program offered through the University of California, Irvine extension program. He teaches a specialized course focused on the development of software for medical devices. He graduated from the University of California, Irvine, in 1974 with a degree in Information and Computer Science.

Schedule

All TSP Symposium 2011 sessions, continental breakfast, and breaks will be held on the second floor of the Georgia Tech Hotel and Conference Center. Lunch will be served in the dining room on the first floor.

Monday, September 19

9:00 a.m.–5:00 p.m.	**Exploring TSP: An Introduction** (Dan Burton and Bill Nichols)
9:00 a.m.–5:00 p.m.	**Introduction to the Accelerated Improvement Method (AIM)** (Tim Chick and Jim McHale)

Tuesday, September 20

7:30 a.m.–8:30 a.m.	Continental Breakfast and Registration
8:15 a.m.	**Symposium Welcome** —Paul Nielsen, Director and Chief Executive Officer, Software Engineering Institute
8:30 a.m.	**Keynote Address**—Capers Jones
9:30 a.m.	**Excellence: Methodology Assists, Discipline Delivers** (David Ratnaraj)
10:15 a.m.	Morning Break
10:30 a.m.	**First TSP Results at Ecuador and Colombia, A Shared Successful Effort** (Hector Gonzalez-Santos and Pablo Henriquez)
11:15 a.m.	**A Survival Guide for Leaders** (Dan Wall)
12:00 p.m.	Lunch
1:15 p.m.	**Updating the Quality Profile for Modern Development Environments** (Noopur Davis and Darryl Davis)
2:00 p.m.	**An Architect's Point of View on TSP** (Felix Bachmann)
2:45 p.m.	Afternoon Break
3:15 p.m.	**Paths of Adoption: A Taxonomy of Pre-TSP/TPI Teams** (David Saint-Amand and Mark Stockmyer)
4:00 p.m.	**The Emphasis Should be on Team, not Software Process** (Robert Musson)
4:45 p.m.	**Daily Wrap-Up**
5:00 p.m.	**Birds of a Feather Sessions**
6:00 p.m.	Welcome Reception Begins

Wednesday, September 21

7:30 a.m.–8:30 a.m.	Continental Breakfast
8:15 a.m.	**Morning Announcements**
8:30 a.m.	**Keynote Address**—Michael Sapenter
9:30 a.m.	**Reporting Project Status to Management** (François Auradon and Noopur Davis)
10:15 a.m.	Morning Break
10:30 a.m.	**Applying TSP for Services: Seven Key Lessons Learned** (Oscar Mondragon and Alan Willett)
11:15 a.m.	**Team Software Process at Adobe Systems: Current State of the Program** (Barbara Spencer)
12:00 p.m.	Lunch
1:15 p.m.	**A PSP Analysis of Defects Injected During Detailed Design** (Diego Vallespir and Bill Nichols)
2:00 p.m.	**Broadening the Ability to Train and Launch Effective Engineering and Service Teams** (Jeffrey Schwalb and Bradley Hodgins)
2:45 p.m.	Afternoon Break
3:15 p.m.	**SEI Interactive Discussions:** **Coaching Teams: The Soft Side of TSP** (Tim Chick and Gene Miluk) **Coaching vs. Dumb Luck** (Jim McHale and Bill Nichols)
4:45 p.m.	**Daily Wrap-Up**
5:00 p.m.	**Birds of a Feather Sessions**

Thursday, September 22

7:30 a.m.– 8:30 a.m.	Continental Breakfast
8:15 a.m.	**Morning Announcements**
8:30 a.m.	**Keynote**—Carl Wyrwa
9:30 a.m.	**A Highly Successful Canceled Project** (David Jolley)
10:15 a.m.	Morning Break
10:30 a.m.	**Changing Software Management Culture from Academic** (Keiichi Katamine and Yoshihiro Akiyama)
11:15 a.m.	**Finding the Estimating Data** (David Malley)
12:00 p.m.	Lunch
1:15 p.m.	**Five Lessons I Learned From an Inspiring Leader** (Daniel Roy)
2:00 p.m.	**Mastering the Coaching of Excellence** (Alan Willett)
2:45 p.m.	Afternoon Break
3:15 p.m.	**TSP on an Architecture-Driven Project** (Luis Carballo and Jim McHale)
4:00 p.m.	**Symposium Wrap-Up and Adjournment**

Abstracts: Monday, September 19

These tutorials are priced separately from TSP Symposium 2011 registration. Stop by the registration desk if you are interested in attending a Monday tutorial. The fee includes the tutorial, continental breakfast, lunch, and breaks.

Exploring TSP: An Introduction

Dan Burton and Bill Nichols, Software Engineering Institute
9:00 a.m. – 5:00 p.m.

In the current economy, organizations are looking for ways to maximize their return on investment. Capers Jones has rated TSP as one of the top methods across small, medium, and large software projects for improving cost, schedule, and quality performance in software development organizations.

Industry data has quantitatively demonstrated that using TSP significantly improves a team's overall performance. This one day tutorial provides an overview of the core TSP concepts and principles and provides the foundation needed to begin to introduce and apply TSP in your organization.

Key topics include
- the concepts on which the TSP is built
- how the TSP improves software development activities and provides positive motivation for engineers and project team
- how to use the TSP to address current and future software needs
- how to successfully introduce and maintain TSP
- what management must do to help their teams be successful

Introduction to the Accelerated Improvement Method (AIM)

Tim Chick and Jim McHale, Software Engineering Institute
9:00 a.m. – 5:00 p.m.

This tutorial is designed to introduce organizational leaders, process improvement champions, consultants, and advocates to the SEI's new Accelerated Improvement Method (AIM). This tutorial concentrates on the concepts and strategies associated with AIM, which is a radical departure from the traditional methods of CMMI implementation, technology transition, and organizational change. It provides a strategic, organizational focus while implementing performance improvements through a tactical bottom-up approach. AIM integrates and leverages established and effective improvement technologies: CMMI, SCAMPI, Team Software Process (TSP), a rapid deployment strategy, and the Six Sigma toolkit. This integration has resulted in a repeatable fast track to high performance.

Key topics include
- overview of the SEI's new AIM
- review of the technologies, tools, methods, and strategies used in AIM
- discussion of AIM deployment
- results that organizations have achieved by applying the AIM strategy

Abstracts: Tuesday, September 20

Keynote: Evaluating and Ranking Software Methods and Practices

Capers Jones
8:30 a.m.

There are dozens of software development methods that include Agile, Crystal development, extreme programming (XP), the Rational Unified Process (RUP), the Personal Software Process (PSP), the Team Software Process (TSP), Rapid Application Development (RAD) and many more. There are also the five levels of the Capability Maturity Model Integration (CMMI) developed by the Software Engineering Institute.

This keynote address provides a method of evaluating methods based on quantitative data. It also evaluates a dozen popular methods and shows their current productivity and quality levels.

Software methods are not a "one size fits all" approach. Some methods such as Agile are most effective for small projects below 1,000 function points. Others such as RUP and TSP excel for larger applications above 10,000 function points.

Methods are evaluated using a multi-criteria approach. The rankings include best practices, good practices, fair practices, neutral practices, harmful practices, and a set of very harmful methods that should be considered to approach professional malpractice if used on important software applications.

Data available to date indicates the TSP is one of the most successful development methods for complex systems due to its focus on quality and measurement.

Excellence: Methodology Assists, Discipline Delivers

David Ratnaraj, Advanced Information Services Inc.
9:30 a.m.

AIS is a strong CMMI Maturity Level 5 company that has consistently produced high quality results for multiple years. AIS' history shows that TSP/PSP processes integrated with CMMI produces consistent and high quality results. These processes provide a strong platform for all new AIS development projects.

This presentation will discuss one project that, despite strong organizational capabilities, faced common concerns:
- The planned size of this project is 340 KLOC. This project will be the largest undertaken by the organization. Will the processes scale up?
- The project uses current technology. There is no organizational historic data for this technology. Will historic data be relevant?
- The team size is planned for 17 with 7 new team members. The organization has not executed projects with this team size. Will the processes scale up? Will team size impact schedule?

The presentation will discuss
- the decisions behind customizing some standard organization processes during the planning process to better fit the project needs
- the various quantitative analyses on results of pilot components and the iterative refinement of processes
- the challenges and changes to team dynamics that needed to be adapted to get new team members to work at Maturity Level 5 quickly and consistently
- the challenges and changes to team dynamics and new role responsibilities that needed to be incorporated to enhance disciplined and efficient working among all team members
- the challenges in dealing with multiple customer stakeholders located in different geographical locations and the adaptive process changes made by the team to ensure all stakeholders' goals were addressed

The team delivered phase 1 of the project on time with 25 defects identified in 115 KLOC of code in user acceptance testing – defect density of 0.21 defects/KLOC. For the team, excellence was not just delivering a high quality product in a timely manner, but also achieving customer satisfaction. Customer feedback for phase 1 indicated that the team exceeded customer's expectations for quality and value. The team's continued success has been primarily due to using well defined processes and a disciplined approach in following them. Large teams could quickly head towards failure if not for these two critical elements.

The presentation will conclude with recommendations on scaling up processes for larger projects and approaches to ensure discipline in work developed by new and experienced developers.

First TSP Results at Ecuador and Colombia: A Shared Successful Effort

Hector Gonzalez-Santos, Kernel; and Pablo Henriquez, Procesix
10:30 a.m.

How do you drive excellence in a TSP implementation within new TSP adopter countries? Procesix and Kernel are convinced that alone we can do so little, but together we can do so much. We are also convinced TSP is the key to achieve a committed team, a defined process, and quality maturity.

In this presentation, we will show our strategy on working together to deploy TSP as a first phase to CMMI, combining efforts to share the benefits these models have for companies to achieve maturity, competitiveness, and excel at their overall performance.

The presentation will include improvements experienced at Bupartech Company, the first TSP implementation in Ecuador, a committed organization that successfully built a self-directed team in a successful TSP introduction. This session will include the first results obtained in Colombia. We will include PSP training results and PSP project and team data. We will also share the effort done in a combined alliance to deploy TSP in other countries in South America.

What are the next steps? What shall we seek to find in a partner and multiply efforts building alliances and sharing efforts on a TSP implementation?

The presentation will show
- Procesix and Kernel working together (successful partnering)
- strategy for implementing and maintaining TSP/PSP
- metrics and performance measures of the PSP-trained team
- overall improvement experienced by the TSP Project: Bupartech Case

A Survival Guide for Leaders

Dan Wall, The Wall Group
11:15 a.m.

It's exciting—even glamorous—to lead others through good times and bad. But leadership also has its dark side: the inevitable attempts to take you out of the game when you're steering your organization through difficult change.

Leading change requires asking people to confront painful issues and give up habits and beliefs they hold dear. Result? Some people try to eliminate change's visible agent—you. Whether they attack you personally, undermine your authority, or seduce you into seeing things their way, their goal is the same: to derail you, easing their pain and restoring familiar order.

This presentation will provide you with practical tools to resist the attempts to remove you and manage your hostile environment. They will help you to continue to propel change forward.

Updating the Quality Profile for Modern Development Environments

Noopur Davis and Darryl Davis, Davis Systems
1:15 p.m.

Watts Humphrey defined the Quality Profile and the Process Quality Index (PQI) as a predictor of component quality in system test and usage. Empirical evidence collected from early Personal Software Process (PSP) experiences showed that components with a PQI greater than 0.4 indicated near zero defects found in system test and in usage. The PQI reflects five considerations of the software development process: 1) the ratio of effort spent in detailed design vs. test, 2) the ratio of effort spent in code vs. personal code review, 3) the ratio of effort spent in detailed design vs. detailed design review, 4) unit test defect density, and 4) compile defect density.

We examined data from more than 100 projects we have coached, and found that the benchmarks specified by the SEI need to be recalibrated for certain types of projects: specifically, commercial software product development. We will share the re-calibrated PQI that we have been using with our teams with some success.

The original PQI did not consider the impact of inspections on component quality. We will share how we have incorporated inspections in the PQI.

The other problem we faced with the PQI as defined by Watts Humphrey is with the unit test defect density and the compile defect density measures: most modern development environments do not include a compile step, and with Test Driven Development (TDD) and automated unit test frameworks, measuring unit test defect density is difficult. The purpose of measuring compile defects was an independent, unbiased method of determining code quality. This usually means the use of a tool. In the spirit of replacing one tool-determined way of determining code quality with another, we encouraged our teams to use static code analyzers. We collected data from 13 projects, and the results show that static code analysis is a valid replacement for the compiler. We will share our results in this presentation. Replacing unit test has been harder: we tried substituting code coverage, but the results were not as conclusive as static code analysis. We will also share our results with trying to find a replacement for the unit test defect density component of the PQI.

An Architect's Point of View on TSP

Felix Bachmann, Software Engineering Institute
2:00 p.m.

Why would an architect be interested in TSP? TSP is for developers, right? And also, architecting is a creative task; this cannot be squeezed into any process. Wouldn't you agree?

Using the Team Software Process (TSP) for designing a system is beneficial not only to plan, estimate, and track the architect's work but also to gain a better understanding in how to support TSP development teams so that they can do their job more efficiently.

TSP is strongly based on providing a measurement framework to gain insights in what is happening in a project. Some of the measurements are geared to implementing code. "Size" is usually measured in lines of code. This does not apply to architecture tasks since there are no lines of code produced. Similar issues arise for measuring quality. Tracking bugs in implementations is a good measure for code quality, but what would be the appropriate quality measure for architecting tasks?

But not everything an architect does is technical work. A big portion of the architect's everyday work is communicating with the various stakeholders. This presentation provides some insights into typical architecting tasks and how those can be estimated, scheduled and tracked. It shows also how the TSP data collected for the technical tasks supports the communication with the stakeholders, such as the product managers, the developers, or even the CEO. Overall, the presentation will show that more transparency into the art of architecting, introduced by using TSP, is not only beneficial for the architect but also for the developers and managers.

Paths of Adoption: A Taxonomy of Pre-TSP/TPI Teams

David Saint-Amand and Mark Stockmyer, NAVAIR
3:15 p.m.

It's easy to state that not all pre-TSP/TPI teams are alike but how different can they be? How do they make their way from the dark ages of ad-hoc process to disciplined superstars? We will discuss the different pre-process team types from the self-actualized champions to the folded-arm curmudgeons. Also covered will be techniques for dealing with the different types of teams, baby-step introduction methods, and a multi-level measurement scale to describe the maturity of TSP/TPI teams.

The Emphasis Should be on Team, Not Software Process

Robert Musson, Microsoft
4:00 p.m.

The TSP is typically seen as a way to build software faster, better, and cheaper. The focus is more often on software process and less on team. However, the true power of the TSP rests with the ability to focus desired organizational behaviors. Team is the key to TSP.

This presentation will focus on organizational factors, known as organizational citizenship behaviors (OCBs), which are affected by elements of the TSP. OCBs are known to be a necessary part of organizations that outperform their peers. Factors such as trust and altruism are directly impacted by the team environment. A model will be presented that relates important elements of an organization and how TSP directly impacts those elements. Various aspects of the TSP are diagnosed from the perspective of building an organization that outperforms rival organizations. The behaviors of several teams will be compared and assessed against the elements of the TSP to determine behaviors most likely to result in high performance. Finally, the resulting improvements to productivity will be presented along with benchmark team data for traditional development teams. From this data, behaviors associated with high performing team will be analyzed.

Abstracts: Wednesday, September 21

Process Improvement from Senior Management's Perspective

Michael Sapenter
8:30 a.m.

CGI Federal has been process-based for the past 20 years or more. The organization has developed processes and tools that have proven to be a consistent set of methods to develop, test, integrate, train, configure, and manage the tactical software systems it creates. The real challenge became how to integrate and improve the existing people skills and experience factors.

The AIM process appeared to provide the right set of people attributes to address the individual estimating, planning, and discipline needs, as well as the integration and balance required for the team communication and coordination. We were trying to answer the question, "How do you really improve the processes without improving the staff?"

With people at the center of the process and the AIM tool sets at your disposal, you can estimate, plan, tailor, balance, and measure improvement in performance. Most importantly, you can address the improvements realized during implementation collectively during all phases of the development process. This is a more natural approach to the reduction in defects, increase in productivity, and support for realistic expectations for project teams. Through process scripts, our standards and forms are implemented using the individual's self-management process. This provides a model for metrics to become a part of each person's continuous improvement program and risk programs to improve the process over the life cycle. The problem of the day and its solution will be clearly in front of each member of the team.

Reporting Project Status to Management

François Auradon, Beckman Coulter; and Noopur Davis, Davis System
9:30 a.m.

How do you report status to management when your TSP team is working on multiple, simultaneous releases, requirements are changing, you have an integrated product team, and management has great expectations about schedule and quality management? Our team struggled for several months to present the right information in the right format to our management. After several false starts, we have now settled on a format that is helpful to both management and to the team, and that can be generated without too much effort from the team planning and quality managers.

We regularly report scope, schedule, and quality status to management. In this presentation, we will share our experiences, the ones that did not work so well, and the most recent ones that seem to be working.

Applying TSP for Services: Seven Key Lessons Learned

Oscar Mondragon, SIE Center; and Alan Willett, Oxseeker, Inc.
10:30 a.m.

The Team Software Process (TSP) is a methodology developed to manage self directed teams that are capable to achieve high performance at developing quality software. The many successes of TSP implementations on software projects has been brought to the attention of managers who are also responsible for other groups; i.e., systems engineering, organizational training, help desk, implementers of software products.

This presentation describes another pilot on how TSP has been used to help service organizations. The TSP was implemented to manage a group of engineers in charge of implementing proprietary software products at customer locations (functionality gap, product configuration, test, install, and train).

The presentation first describes the preparation work for the launch—understanding the organization, explaining benefits of TSP in their environment, building the sponsorship, defining the scope, sketching the service processes, and providing the training. Second, it describes the points of inflection in the launch steps—defining the strategy to handle the work and the organization, defining the conceptual design, customizing services processes, customizing size measures, customizing defect data, and defining the quality plan. Third, it describes the points of inflection on tracking the project's progress—handling changes and dealing with the right abstraction for progress. Fourth, it describes the project results—team performance, measurement, improvement, benefits, changing the team's way of working, and the overall improvement experienced in the service project. Finally in the lessons learned, it describes how to handle TSP implementation problems such as denying the reality for not fulfilling customer compromises and stepping on dangerous or inadequate organizational practices.

Team Software Process at Adobe Systems: Current State of the Program

Barbara Spencer, Adobe Systems, Inc.
11:15 a.m.

This presentation will cover the experiences of a team at Adobe Systems that has been using the Team Software Process since 2009 as part of a three year project. The team is within a large product group. This session will share the team's performance, improvements, and benefits so far from using TSP, and will discuss how TSP has impacted their work from the perspective of team members, team leader, and their senior management. This presentation will also cover a brief history of the overall TSP program at Adobe up to the present, which is currently just one team doing TSP. The presentation will explore the current context and reasons for the size of the TSP program at Adobe, as well as the substantial benefits that Adobe Systems has gained so far from doing TSP.

A PSP Analysis of Defects Injected During Detailed Design

Diego Vallespir, Universidad de la Republic; and
Bill Nichols, Software Engineering Institute
1:15 p.m.

Because the PSP is a personal process, what happens using PSP remains personal. So therefore, the improvements to the process are personal. We all know, however, although we are different, we are in many ways the same. We demonstrate this in PSP training by presenting aggregated class results to the class. It is through the statistical analysis of the aggregated results that we demonstrate the effectiveness of PSP and explore the software development process. However, this has not yet been done with recent versions of the course. Because of limits in the data availability, some aspects of the improvement have been left unexplored.

A new source of data is now available through the SEI's Blended Learning system. This system has also collected data in greater detail than was available previously. We will now leverage that new data source to better understand the typical expected improvement from the PSP course. We have begun statistical analysis by analyzing the programs of all students using C, C++, and Java and have completed the final exercise PSP II.

In this study, we present an analysis of defects injected during the design phase in PSP exercises 6, 7, and 8. We use familiar PSP analysis, but limit consideration to the defects injected in design. We observe how defects injected during design escape into each phase of the PSP and how the cost to remove is affected by defect. We describe the different defect types injected during design, and how these defect types correlate with the find-and-fix time. From this analysis we draw some conclusions about defect behavior through the process. Finally, we study the defects that escape into Unit Test.

Excellence in PSP means efficiently driving toward zero defects escaping into Unit Test. We analyze improvement opportunities to achieve even better process yields by studying the design defects through the PSP phases.

Please see page 19 to read the paper associated with this presentation.

Broadening the Ability to Train and Launch Effective Engineering and Service Teams

Jeffrey Schwalb and Bradley Hodgins, NAVAIR
2:00 p.m.

The TSP for software engineering teams does well due to training provided by the PSP for Engineers course. However, starting a team that provides other products and services such as systems engineering or product support has been challenged by the lack of depth in comparable training for these other areas.

This approach interlaces personal process and just-in-time team member training course modules along with the application of process modeling techniques. This means the team being trained also performs process modeling on the activities used to actually provide their products and services. These processes are then folded into the team training as examples. They are also applied in the actual launch as life-cycle models for the products and services being planned. Generation of team processes during training allows the team to enter the launch with a strong connection between planning the work and knowing how the actual work is performed.

Another benefit of this approach is the attention paid to quality. This is accomplished by introducing the team to the notion of mistake injection and removal phases in their newly defined processes. This begins during initial modeling of the processes and is then repeatedly emphasized through to the review of the process during the third meeting of the launch.

Finally, this defined set of process activities lends itself to the ability to generate an environment that uses the appropriate nouns and verbs so that a team hits the ground running from the first day of the launch.

Coaching Teams: The Soft Side of TSP

Tim Chick and Gene Miluk, Software Engineering Institute
3:15 p.m.

When a team's members share a common goal, know how they are going to reach the goal, and are able to work and support each other, they can achieve their best performance. Working in a team setting like this can be a motivating, satisfying, and rewarding experience, but it is also rare.

Not every group of people working together behaves like a team. Because nearly all development work involves groups of people working together, nearly all development work can be improved by having the capability to quickly build jelled, high-performance teams. This is the responsibility of the TSP Coach.

In this interactive we will define the coaching role and discuss methods for building and sustaining high-performance teams.

Coaching vs. Dumb Luck

Jim McHale and Bill Nichols, Software Engineering Institute
3:15 p.m.

Most active coaches have been at both ends of the spectrum of "coaching luck." The senior manager walks in with one of the best meeting 1 presentations ever—or no slides at all. The team of rookies cruises through the launch like they've been doing it for years—or the team of veterans treats every process step like it's the first time they've done it. The review team uses checklists like scalpels—or reads them like they are translating Sanskrit.

While some of the successes can be attributed to fortunate circumstances, Louis Pasteur's famous observation is most often applicable: "Chance favors only the prepared mind." This session presents some common coaching scenarios, from organizational introduction to individual guidance, and discusses the roles of prescribed preparations like the pre-launch checklist, and less obvious ones, like building a relationship with the management team, in ensuring that your coaching experience is prepared for fortunate happenings at all times.

Abstracts: Thursday, September 22

Keynote: Medical Device Software Development

Carl Wyrwa
8:30 a.m.

Developing software for medical devices is a very challenging undertaking. The quality of the software must be extremely high to meet the demands of providing safe and effective products involved in the critical care of patients. Medical devices are regulated by the United States Food and Drug Administration and by similar agencies in other countries. This requires constant focus on meeting all of the regulatory requirements established for medical device software development on a day-to-day basis.

For more than three decades I have lived in a world of constant focus on the quality of the software being produced and constant oversight of the processes used to develop the software to ensure that there are no missing steps that might introduce regulatory compliance gaps. I have spent a tremendous amount of my time doing what is necessary to make sure these go well.

Working with our TSP projects is a delight. With the TSP in place, meeting these challenges is now becoming second nature to the team. We will now be able to focus much more on delivering full feature sets and innovative solutions to our customers and the patients they care for.

A Highly Successful Canceled Project

David Jolley, Hill Air Force Base
9:30 a.m.

In 2004, the United States Marines Corps approached a CMMI Level 5 group in the United States Air Force for software support of their amphibious Expeditionary Fighting Vehicle. Due to resource constraints, this team at Hill Air Force Base in Utah staffed the team with technical experts who, unfortunately, were not all familiar with CMMI. However, the team was part of an organization that had embraced PSP and TSP and was able to provide a highly capable TSP coach who introduced TSP concepts that allowed the team to rapidly implement high maturity processes and practices. Over the next several years, the team grew from a dozen team members to over a hundred and developed more than one million lines of embedded software code as well as hundreds of thousands of pages of requirements, designs, and test procedures. Not only did this team release engineering builds on schedule in a rapidly changing development environment, but they consistently delivered high quality software with system test defect densities below 0.2 defects per thousand lines of code. The final version of software had a delivered defect density of only 28 defects per MILLION lines of added, modified, and deleted code.

The Hill EFV team used TSP across all coding aspects, including Mobility, Power and Auxiliary, Controls and Displays, Fire Control, and even support products such as Test Tools. TSP concepts were also used for System Software Design, System Test, and Command and Control Teams. They experienced great success with a multi-team approach to TSP launches and were also able to modify the launch and tracking processes to account for rapidly changing requirements; this approach eventually brought some peace of mind to a frantic customer.

While the EFV project was canceled in early 2011 due to total cost of ownership, the Hill team's effort was considered enormously successful and the team members are now taking leadership roles on new projects throughout the Hill group, bringing the practices of TSP with them.

Changing Software Management Culture from Academic

Keiichi Katamine, Kyushu Institute of Technology; and Yoshihiro Akiyama, Next Process Institute Ltd.
10:30 a.m.

This presentation describes the implementation methodology of PSP and TSP for graduate students and how engineering skills of students are improved in terms of planning and quality. Issues and future work are also described.

Based on a cooperation agreement with CMU/SEI supported by the Ministry of Education, Culture, Sports, Science and Technology of Japan, Kyushu Institute of Technology has introduced PSP and TSP into its curriculum for graduate students to produce responsible software engineers who acquire practical engineering skills for software development. First, three faculty members acquired the certification of PSP instructor. The training courses PSP-I and II of PSP for Engineers have been conducted as regular courses since 2007, and 25 graduate students took the course in total. The TSPi course, the educational version of TSP, was also started in 2010, and the first TSPi team of three students successfully finished their project.

Major issues of incorporating PSP into a yearly class schedule are the most effective allocation of lectures and instructions to the limited time slots of class and programming skills of students required to complete PSP assignments. 100% of students completed PSP-I in 2010 by certainly confirming the understanding of requirements, the soundness of conceptual design, and the correctness of estimation process in the planning phase, while 43% in the first year.

Quality improvement is clarified by the trend curve of defect density and test time through the courses. The average defect density in unit testing is gradually reduced through Assignments 1 to 8, and 26.7 defects per KLOC for Assignment 1 was 7.7 for Assignment 8. The percentage of test time in total development time indicates the same improvement result.

Three students who completed Assignment 6 at least built a TSPi team, and challenged the change counter example. They completed only one cycle even though they chose the development strategy of two cycles. One reason for the schedule change is high defect density, 42.6 defects per KLOC, in unit testing. However, they successfully finished the first cycle without any defects in system testing because they decided to rework the design in accordance with the stated guidelines of TSPi.

Because insufficient skills of coaching a TSPi team are also clarified, it is planned to improve the coaching skills through TSP coach training and mentoring.

Please see page 12 to read the paper associated with this presentation.

Finding the Estimating Data

David Malley, Naval Oceanographic Office
11:15 a.m.

This presentation will talk about our project's journey in trying to figure out how to collect and organize historical data in a way that is useful for future estimates. This presentation will give examples of original approach and discuss the issues and drawback associated with it. Then we will talk about the current approach being used, which includes the use of both size and effort based proxy tables, based on the available historical data. We will discuss the full software development lifecycle, from requirements elicitation to product delivery and provide the associated estimation variables.

Five Lessons I Learned from an Inspiring Leader

Daniel Roy, STPP, Inc.
1:15 p.m.

From the many lessons Watts Humphrey taught by example, I have selected five major threads that have impacted my own professional life.

The importance of true commitment. Anecdotes are used to show why total commitment to rather simple quality principles are at the center of Watts's work from the essence of CMMI Maturity Level 2 to the discipline of PSP/TSP.

Discipline is not a bad word. Some psychology is in order to understand why some are more predisposed to a regimen to improve than others. But no matter the predisposition, Watts's work shows how to make discipline a habit for improvement. Level 5 is about personal commitment and discipline. Anecdotes are used to show the impact of disciplined data analysis and commitment to quality on business results.

Let nothing stop you. Anecdotes from my own exam to become a PSP instructor and vignettes from Watts's life are used to show that we succeed when our desire for improvement is stronger than our fear of failure.

The renaissance is over. Watts recognized at the end of his life that genius, outrageous commitment and even extraordinary achievements are not sufficient to have an impact. It takes many teams and collective effort. This may be his most important legacy.

Finally, the principles behind AIM are used to show the impact of Watts's ideas on the present and future of systems engineering.

Mastering the Coaching of Excellence

Alan Willett, Oxseeker, Inc.
2:00 p.m.

After being part of more than 200 TSP launches in the last 12 years, I have been continuously amazed about how much more I know after each experience. This talk will cover the lessons I have learned that will provide the highest leverage to the session participants. The lessons will be highlighted by examples from experiences I have had with TSP teams.

TSP Coaches and team leaders who adapt these ideas will accelerate their growth journey in leading and coaching teams to excellence.

TSP on an Architecture-Driven Project

Luis Carballo, Bursatec; and Jim McHale, Software Engineering Institute
3:15 p.m.

From August 2009 to June 2011, the SEI worked with la Bolsa Mexicana de Valores (BMV)—the Mexican stock exchange —on a project to build a new online securities trading engine that will replace Mexico's three existing systems that trade stocks, futures, and derivatives. BMV chose two SEI technologies to help guide the project: the Team Software Process (TSP) and Architecture-Centric Engineering (ACE).

Beginning in July 2011 the system enters system test, and live validation testing, running in parallel "shadow mode" to the existing system, is scheduled for September 2011. This talk presents selected project results, process data, and other interesting project information including the starting conditions, architectural quality attributes (both desired and as they stand), the project timeline, cycle-by-cycle process data summaries, process issues surrounding the use of ACE practices, tailoring considerations in using ACE on a TSP team, and preliminary recommendations for using ACE on future TSP projects.

Changing Software Management Culture from Academic

Keiichi Katamine, *Kyushu Institute of Technology*
Masanobu Umeda, *Kyushu Institute of Technology*
Masaaki Hashimoto, *Kyushu Institute of Technology*
Yoshihiro Akiyama, *Next Process Institute Ltd.*

1. INTRODUCTION

Quality and effective management are essential, in all senses of the word, to be successful in business and to ensure life in a safe and dependable society. Software development is no exception. The software process is the key behind high-quality software and successful projects, as it can include and integrate the development of system technologies, and the utilization, application, and transfer of these technologies in a straightforward way. Such best practices are demanded in the industry.

In the latter half of the 1980s, Dr. Watts Humphrey investigated the industry's software problems and found that most of the problems encountered in software development projects could be solved if software engineers acquired knowledge of the software process and its development. As a consequence, he developed the Personal Software Process (PSP) [Humphrey 95] [Humphrey 05] for software engineers and management to learn best practice at an individual level, and the Team Software Process (TSP) to successfully lead software projects by building and managing a team comprised of members trained in PSP.

The Kyushu Institute of Technology in cooperation with the Carnegie Mellon Software Engineering Institute (SEI) has begun new courses in software processes. These courses are based on PSP, TSP, and TSPi, which is an academic version of TSP [Humphrey 99]. It is our challenge to identify all current software practices at universities in Japan to find a better way of developing these practices and to demonstrate this performance from an academic viewpoint. This paper describes the design and implementation of a process education course for graduate students using PSP, TSP, and TSPi. Section 2 describes the trend in process education. Section 3 describes the design and implementation of process education courses for graduate students. Section 4 summaries the course results. Section 5 discusses any remaining issues.

2. BACKGROUND

While one organization can continue to produce high-quality software, others continue to produce low-quality software [SEI-MS 04][HUESCA 10]. Even if a low-quality performance organization attempts to make the transition to a high-performance organization, such development is difficult and takes a very long time, say four to five years. Take the example of an academic laboratory, their efficiency and progress on research may deteriorate, and the transition to a high-performance laboratory does not happen effortlessly because their work becomes more complicated. A similar problem is encountered in software engineering studies. Top students who have completed comprehensive degrees often fail to use or apply in their jobs what they learned at university. They will eventually forget all that they learned in their studies.

"Culture" creates the norms that characterize organizations, teams, and individuals, particularly the performance of these entities, via various attributes such as beliefs, knowledge, values, rules, behaviors, capabilities, and various inheritances from previous generations [Dictionary]. As software development is knowledge work, the culture of an organization is deeply rooted at individual and team levels, and the issues described above are viewed as culture-related problems.

A cultural change can occur within a software organization when a set of appropriately selected attributes are changed consistently, collectively (or cooperatively), and uniformly. These changes must occur within every individual, leader, and management level of the organization. It is necessary that you know how, when, and where such changes to the relevant attributes are accomplished and controlled consistently, collectively, and uniformly at individual and team levels throughout the software process. To ensure that a change in software culture occurs, it is important for an organization to integrate uniform process principles at both individual engineer and team levels.

A change in culture for software engineers starts at academic school. As many years will be spent training in and eventually working on software technologies and processes, any improvements in software activities for individual students and student teams should be taken to ensure the minimization of reworking, un-learning, and re-learning.

2.1 Trends in Process Education

Until the early 1990s, software engineering training and education focused on software development techniques such as requirement analysis, object-oriented analysis and design, programming, reviewing, and testing. Minimal research efforts were put into software processes that could improve software quality. Then, organizations began to look at improving software processes with the release of process models or process standards such as CMMI, ISO, and IEC, heralded as best practice methods. However, improvements at the organizational-level were very slow and transient.

The SEI introduced official training courses based on operational processes: PSP for engineers and instructors, and TSP for engineers, coaches, and managements, which covered training appropriate at all organization levels. Further benefits included 1) "a process system" that engineers need to know, understand, and follow with fidelity to produce high quality products, 2) rational management using data and facts throughout the work lifecycle, including planning, and 3) a strong mindset to produce high-quality work products before testing.

PSP training courses work to change the culture of engineers in a straightforward manner, a result that has been observed at every course. The PSP instructor plays a coaching role to the course students so that every student learns key concepts, obtains basic skills that they can apply in their work, and process disciplines. It is essential that the PSP instructor is an expert in the process and engineering.

The TSP coach helps a TSP team and organization to improve its performance. Before embarking, a TSP coach needs to determine if the TSP team is really capable, i.e., will it be able to commit to the expected team goals according to the requirements of the sponsor

and customers regarding time, cost, skills, and quality? Management training is key to aligning the motivation of the engineering team with the project goals.

2.2 PSP and TSP Training in Japan

Dr. Watts Humphrey visited Japan and presented his talk on PSP and TSP at Wasada University on June 16, 2000. This was the catalyst. Teaching staff and several industry professionals then formed a group called the "PSP Network," and they promoted PSP education using the text *Introduction to Personal Software Process* [Humphrey 97] at several Japanese universities and companies. In 2003, 10 software engineers from JASPIC (Japanese Association of Software Process Improvement Consortium, established in 2000) tried TSPi and experienced reasonable success after studying *A Discipline for Software Engineering* [Humphrey 95].

The first SEI-authorized PSP course for engineers was provided by IBM Japan 2003 for five IBM software engineers in 2004. The course was successful, and two people were trained as PSP instructors. Two additional companies held courses, one course was attended by 10 software professionals and the other by 12 engineering graduates. The graduate course was especially successful and it achieved a class average process yield of 75%. To date, software engineers and managers from 11 companies have taken PSP- and TSP-related courses and/or training courses such as PSP for Engineers, TSP Executive Strategy Seminars, Managing TSP Teams, and Leading a Development Team. In addition, as interest grows, TSP coach training courses have been held. Furthermore, Aizu University [AizuUniversity 09][AizuUniversity 10], Shinshu University [KAIYA 01], and the Advanced Institute for Industry Technology have also provided either PSP for Engineers pilot classes or three days seminars on PSP and TSP.

3. INTRODUCTION TO PROCESS EDUCATION

3.1 Collaboration with CMU/SEI

In mid-2006, the Faculty of Computer Science and Systems Engineering of Kyushu Institute of Technology (KIT) decided to introduce PSP and TSP training, and in early January 2007 a three-year KIT-SEI collaboration program was accepted [AKIYAMA 10]. To increase faculty members' interest in the program, keynote presentations by SEI representatives were scheduled. Dr. Paul Nielsen, the director and CEO of the SEI, was invited to the opening ceremony at the Center for ICT Education in May 2007. His keynote talk focused the importance of software technology and software process to ensure successful technological development [NIELSEN 07]. Dr. Linda Northrop was also invited and gave presentations on two topics: Software Product Line Management and Ultra Large-Scale Systems, which were new issues for faculty members and students alike [NORTHROP 07].

We also planned faculty workshops for the summer of 2007. There were 22 attendees: 13 teaching staff (including professors) from KIT and 9 from another 7 universities. One PSP/TSP representative was invited from the SEI and the workshop was useful in offering the attendees knowledge and understanding regarding the PSP and TSP programs to be taught at university.

Three workshops were planned over three years, conducted by SEI representatives to provide overviews on PSP principles and technologies, the potential effect on education, TSP principles and how they are applied to graduate students, systems engineering, and TSP to determine what we need to enhance current software system education. A three-day CMMI course was provided in late 2008 for selected faculty members who were interested in the subject. These workshops and courses also help to advertise relevant current and future events.

After the training of three PSP instructors at KIT, two pilot classes were conducted during 2007 and 2008, with five students completing the PSP course. When Dr. Nielsen was invited to KIT and presented PSP course certificates to the five master course students on September 15, 2009, accompanied by Dr. Shimomura, KIT president, and Dr. Oie, dean of the school. This was a great event for all PSP students and instructors.

The impact of these events has been significant, as illustrated by the following four examples: 1) an increase in the number of students taking the PSP course (3 students at the first PSP course in 2008 and, now in 2011, there are 13 top-level PSP students); 2) an industry version of TSP for a PBL course is included in the graduate school curriculum; and 3) "Team Member Training" has been introduced into undergraduate courses, in addition to "Introduction to Personal Process."

3.2 Course Design

To follow is an outline of the PSP/TSPi training course designed at KIT for graduate students.

3.2.1 Context of course design

In the late 1990s, several of the authors of this paper considered the challenge of how to use knowledge-intensive information communication systems to solve real-world integration problems of environment, society, industry, and personal life. In 2002, the Department of Creative Informatics was established to run a new course at the Graduate School of Computer Science and Systems Engineering at KIT, for advanced education.

However, as undergraduate students only learn fundamental programming using Java and C languages, data structure and algorithms, they have no real experience of projects or large-scale software development, which is required for solving real-world problems with information communication systems. Therefore, the Department of Creative Informatics was designed to provide further education in project management, software architecture, systems and software engineering, business analysis/modeling, and business re-engineering. PSP/TSPi were introduced in the above context.

3.2.2 PSP

PSP is the basic process that enables a software engineer to learn the fundamental process that is followed throughout the development lifecycle of a module or component of a larger software design. PSP enables every student to improve their process via course assignments and using process data recorded by the Process Improvement Proposal (PIP) regarding defects, time, and size.

PSP provides basic training on simple software processes and best fit for academic students, industry software engineers, and software managers. This course includes two *basic* parts: PSP-I (planning) and PSP-II (quality). PSP-I training emphasizes the process discipline, measurement, estimation, and planning, and this is provided for in the first quarter of the first master course school year. PSP-II establishes engineers' disciplines for quality management and is offered in the second quarter of the school year.

3.2.3 TSPi

Large-scale software is usually developed by teams. TSP is used essentially in the development of software and each team member defines their own individual and team processes to accomplish the team goals. The integration of individual and team processes is extremely important to ensure the consistent, collective, and uniform alignment of the members' beliefs, knowledge, values, rules, behaviors, capabilities, and those inherited from the previous team's efforts, and to ultimately accomplish the team goals.

KIT uses TSPi. While TSP introduces team building, project planning, project tracking, and risk management, it is too comprehensive for academic student training; thus, TSPi [Humphrey 99] was designed specifically for university students. It uses multiple development cycles, with each cycle comprised of launching stages, strategy, planning, requirements, design, implementation, testing, and a postmortem. The TSPi course is provided during the third and fourth quarters. Students who complete PSP-I are also accepted into the TSPi course.

4. COURSE MANAGEMENT AND RESULTS

4.1 PSP

4.1.1 Managing the PSP Course

The main challenge of introducing PSP into graduate programming is how to resolve SEI course requirements within the design of the graduate school curriculum framework.

1. Course schedule structure

PSP for Engineers consists of two course parts, PSP-I and PSP-II. Students spend one week each on PSP-I and PSP-II. The general structure of a course-day is lectures in the mornings and assignments in the afternoon. The assignments given on the fifth days of the PSP-I and PSP-II courses are interim and final reports, respectively.

The graduate school curriculum provides, as one unit, a total of 22.5 hours (15 periods of 1.5 hours) for one quarter, which is assigned to either PSP-I or PSP-II. The challenge is how to marry the required course hours with the limited class hours per unit.

2. Insufficient programming skills and experience

If a student does not have enough programming skills and experience, s/he will not advance beyond the first PSP assignment, which requires designing and implementing a simple data structure and operation for a linked list. This makes it difficult to estimate the program length at the very beginning. As shown in Table 1 with regard to the evolution of teaching practices from 2007 to 2011, we identified an appropriate course pace and length via confirmation of class progress, process performance, and quality improvements.

After the instructor-qualifications of the authors in 2007, PSP-I was conducted in the SEI format, i.e., an intensive one-week course, and we found that some students found it quite hard to keep up with the course pace. Only 43% of **students completed all five assignments (four program assignments and one report assignment). At that stage, PSP-II lectures were held every two days, ensuring that the students had at least one day to complete their tasks.**

In 2008, to increase course completion rates, lectures were held only once a week. This gave students seven days for each exercise. To resolve the issue of students with insufficient programming skill and experience, we introduced a pre-course program of design and implementation, aimed at raising the students' skill-base. The completion percentages for PSP-I and PSP-II increased to 75% and 100%, respectively. However, these high rates were also in part to the fact that students who realized they were lacking in programming experience during the pre-course program withdrew.

Since 2009, there have been no **pre-course** programs and all students who want to take the course are accepted. We now provide a reasonable length of time for a plan review to ensure every student has an accurate understanding of the course requirements and of the conceptual design and estimation processes. This is to maintain the students' motivation and to avoid withdrawals caused by rework. PSP-I completion rates increased to 80% in 2009. In 2010 and 2011, PSP-I completion rates sat at 100% using a similar approach.

Year	Part	Method
2007	PSP-I	Three-hour lectures over five consecutive days
	PSP-II	Three-**hour lecture once every two days**
2008	PSP-I	Six-hour pre-course exercise, three-hour lecture once a week
	PSP-II	Three-**hour lecture once every two days**
2009 to 2011	PSP-I	Three-hour lecture once a week and plan review several days after the lecture
	PSP-II	Three–hour lecture once a week and plan review several days after the lecture

Table 1: Course management for PSP-I and PSP-II

4.1.2 Performance of PSP Courses

This section outlines software quality and productivity from the 25 course graduates of the classes from 2007 to 2010, including interviews with those students.

Software quality can be evaluated by (unit) test defect density. Figure 1 shows the quartile graph for the test defect density of the classes. The horizontal axis shows the assignment numbers and the vertical axis shows the test defect density. The first, second (median), and third quartiles show decreasing densities as the assignment number increases. The difference between the first and third quartiles decreases too. The first and third quartile lines show the progress of the best and the worst performances in software quality, respectively. The best performance at the beginning of the course is worse than the worst performance at the end of the course. The design and code

reviews by checklists and verifications were introduced in assignments 5 and 7, respectively, and show **the effectiveness** of university education.

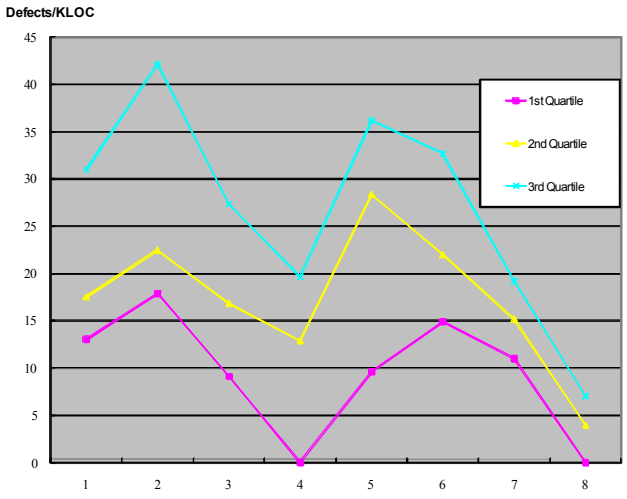


Figure 1: Transition of test defect density

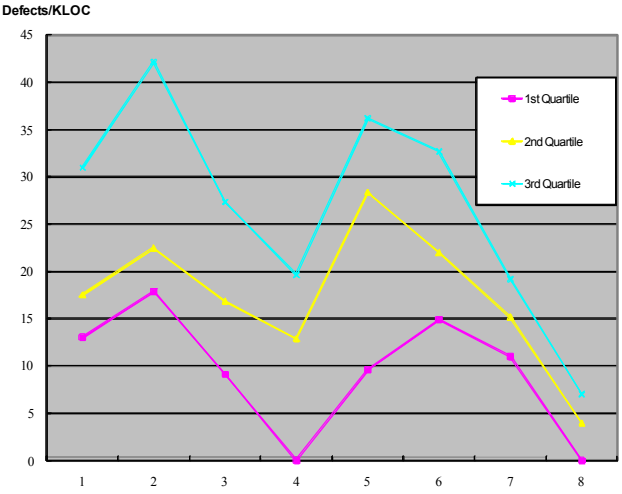


Figure 2: Transition of productivity

Figure 2 shows a quartile graph for productivity performance. Productivity decreases until assignment 6 and then steadily increases to assignment 8. The reason for this is, first, because the students have learned programming practices using disciplined software processes, and second, from assignment 6, the same process from PSP2.1 is followed and the unit testing time decreased. While the productivity shown in the reported SEI class data does not change significantly, the level of productivity doubles in the classes and is very significant.

Moreover, we observed via interviews with the PSP students that they understood the following

- how to make a plan
- the importance of design and review
- defect reduction by recording a defect log
- completion of software development as planned by reducing time variations in defects
- process improvements by using process data
- performance differences between themselves and other students who have not studied PSP

As mentioned above, most students independently recognized the significance of the PSP course by enrolling for the course.

4.2 TSPi

4.2.1 Managing the TSPi Course

The main issues regarding the introduction of TSPi as a graduate course are as follows.

1. Course schedule structure

The graduate curriculum allocates a unit class time of 45 hours, which is three hours x 15 periods. The main issue is how to make the most effective use of the limited class time.

2. Students' lack of experience regarding team activities

Most students at graduate school have very little **industry experience or experience with team** exercises. One important issue is communication among students. However, most students have been assigned a standalone task and small-scale software development, and only a very small number of students have experience with group activities such as school festivals, part-time jobs, and community associations.

3. Lack of TSPi teaching experience

We all had experience in mid-scale software development projects. However, regarding TSPi study, we had no experience and, therefore, a faculty workshop was conducted in the summer of 2007 [CANON 07]. Thus, we gained the relevant TSPi knowledge. Since then, we have received training to teach the entire TSPi course and class implementation.

The first class was conducted using TSPi DEV scripts. We have since improved the teaching method and this can be confirmed by looking at the team progress. An overview of the TSPi course was provided at our first class, and a TSPi student team made a presentation on the project results from the previous class. A basic TSPi process with two cycles was considered appropriate for the 13 class periods, while the DEV script suggested three cycles in 15 class periods. We consulted the team for its opinions, with the team choosing the two-cycle approach. In the end, however, the team **completed** only **one cycle of the 13 class periods for a number of reasons, including** the temporary withdrawal of a member, a misunderstanding of the software requirement specifications, and the re-design and re-coding of the modules according to the software quality review, which was done after the unit testing. Despite this, the team achieved success with zero defects in the cycle's system testing. This is because the team followed the TSPi process faithfully. A presentation of the team result was evaluated by an audience of faculty and industry members. The following section shows the process data in detail.

4.2.2 Performance of the TSPi Team

The performance of the team is discussed in terms of software quality, schedule, and role evaluations based on the process data of the TSPi team 2010. Three students completed Assignment 6 at least, and wanted to try the TSPi project. They made up one TSPi team, and selected the change counter need statement.

Software quality is evaluated by defect density. The team achieved zero defects in the system test. The team successfully completed cycle one. However, approaching the system test, the team found that the defect density of one of the modules in the unit testing was 42.6. This was not a good-quality indicator and the team decided to rework the design of the module before moving to the next step, in accordance with TSPi guidelines.

Figures 3 and 4 show the review rates of the other modules and the defective module (which shows inadequate review of the defective module), respectively. These problems arose from an inappropriate checklist and an incorrect understanding of how to handle multi-byte characters. The team decided to use C language for this project while some used Java in the PSP training. A further team member used a Java checklist for the C program. This increased the defect rate. In addition, the team misunderstood the specifications regarding the software libraries they used. The team reviewed the checklist and studied the software libraries for the multi-byte characters.

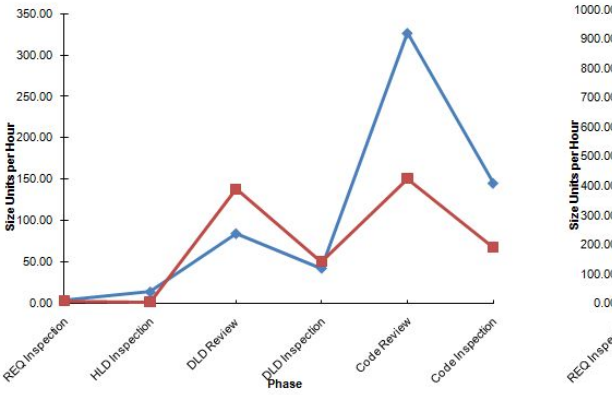


Figure 3: Inspection and review rate for system

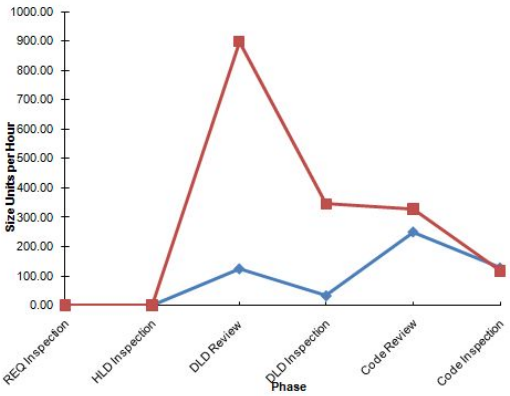


Figure 4: Inspection and review rate for the effective module

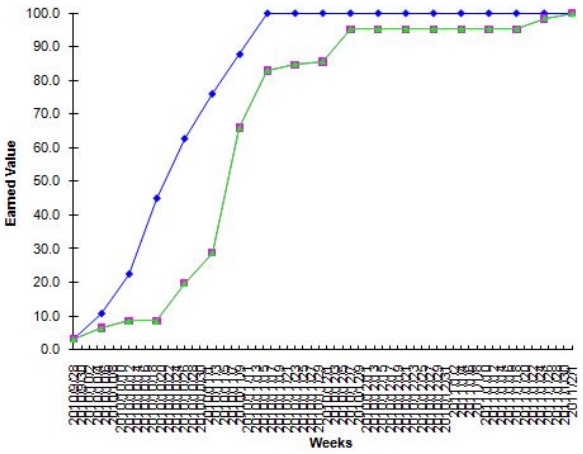


Figure 5: Cumulative planned value and earned value for system

Figure 5 shows the entire EV graph, where we find the occurrence of two delays. The first occurred during the launch phase to the requirements phase. Students have almost no experience in understanding and documenting the requirements that the team needs to address. For instance, they did not know about IEEE Standards 830 and did not understand what and how they were meant to describe the software requirements specifications (SRS). The second instance was the implementation phase, which starts from the detailed level design (DLD) and ends at the unit test (UT). Many defects were included because the specification was difficult for the team as C was selected. The team completed cycle one of the two cycles selected for the development strategy.

The following is a list of some of the feedback given at the role evaluation and the interviews with the TSPi team students.

- Lack of communication caused late responses with the temporary withdrawal of a member due to health problems.
- A team member did not adequately plan for the workload to accommodate the multiple role managers assigned by the team.
- The Configuration Control Board (CCB) was not implemented and risk management was inadequate.
- They spent a lot of time on documentation and management, more than the time spent on development.
- The TSPi tool is efficient in team management.
- There is no manual for the TSPi tool, and it works differently depending on the version of Microsoft Office used.

As mentioned above, most students independently understand the significance of the TSPi course, as illustrated by the fact that they enrolled in the training. In addition, TSPi is an efficient tool for process education in graduate schools, to experience the lifecycle of software development. Especially for students with no industry experience, they can begin with TSPi at school and then advance to TSP. Moreover, the benefits of the TSPi process are assisted by prior knowledge of software systems and architecture, project management, and business system modeling.

5. DISCUSSION

5.1 New Course Design

Subsection 3.2.1, "Context of course design" describes real-world problem solving. The education courses of Department of Creative Informatics cover advanced software systems, systems engineering, business analysis/modeling, and business re-engineering, and as such we can design a real-world project-based learning course by integrating the results of PSP/TSPi training and the education courses.

5.2 Curriculums for Undergraduate Students

It is desirable to introduce PSP training courses to upper-undergraduate students after they have acquired programming knowledge and skills through fundamental software engineering. These courses are vital for students with **engineering software development** and process **skills, to operate in a knowledge-based society**.

6. conclusion

"Culture" shapes the norms that characterize an organization, particularly performance, via beliefs, knowledge, values, rules, behaviors, capabilities, and even inheritances from previous generations. Cultural change within an organization is difficult and requires changes in the relevant attributes at all levels (individual, team, and organization), changes that need to occur consistently, collectively, and uniformly. PSP training is effective and appropriate for all engineers to help establish the fundamental process skills necessary for the lifecycle of quality module development. In addition, PSP training will consistently, collectively, and uniformly align their understanding of the fundamental methods of software estimates, scheduling, quality planning, and monitoring. TSPi is designed for team situations, enabling groups to work together effectively by monitoring their performance using recorded process data and to enable the meeting of team goals to ensure team success. It is of note that these measures are not often included in process models and process standards, or even in best practice descriptions.

Process training using PSP and TSPi courses was introduced in the Department of Creative Informatics Kyushu Institute of Technology's graduate school and instantly showed promise as a means of changing software management culture away from its academic roots. The PSP course has now been delivered for five years (starting 2007) and the results have shown significant and consistent improvements. The first TSPi team came together last year, and while they only completed one cycle (because of several unforeseen problems), they also achieved zero defects in the system test. This was no accident.

In addition, TSPi clarified what knowledge the students need to supplement the process data. TSPi courses will work better if students are familiar with the related topics such as advanced software systems, systems engineering, business analysis/modeling, and business re-engineering. In the near future, we intend to introduce a new course that will allow graduate students to develop their own experiences and views on software management, a course that will offer useful contributions to the software industry.

7. REFERENCES/BIBLIOGRAPHY

[Humphrey 99]
Humphrey, Watts S., *Introduction to the Team Software Process*, Addison-Wesley, 1999.

[Humphrey 05]
Humphrey, Watts S., *A Self-Improvement Process for Software Engineers*, Addison-Wesley, 2005.

[Humphrey 95]
Humphrey, Watts S., *A Discipline for Software Engineering*, Addison-Wesley, 1995.

[Dictionary]
TheFreeDictionary, *http://www.thefreedictionary.com/culture.*

[Humphrey 97]
Humphrey, Watts S., *Introduction to Personal Software Process*, Addison-Wesley, 1997.

[MSE]
Software Engineering Master Programs, *http://mse.isri.cmu.edu/software-engineering/web1-Programs/index.html*.

[HILBURN 00]
Hilburn, Thomas B., *Teams need a process*, ACM SIGC SE Bulletin, Vol. 32, Issue 3 , 2000.

[NIELSEN 07]
Paul Nielsen, *Software Engineering and the Performance Improvement World: A Personal View, May 10th, 2007.*

[CANON 07]
Bob Canon, *Faculty Workshop at KIT, Aug. 7th – 12th, 2007.*

[NICHOLS 08-10]
William Nichols, *Workshops at KIT on PSP and TSP, Mar.23rd-24th,2008, Mar.8th-10th, 2009, and Feb.6th-8th, 2010.*

[NORTHROP 07]
Linda Northrop, *SPL:Reuse that makes Business Sense,* and *Impact of SCALE: Ultra Large Scale System, Sept. 7th, 2008.*

[HUESCA 10]
Huesca, Agustin M., *Best Practices for TSP Implementation on Outsourced Application Development Projects,* Keynote of SEPG-NA, 2010.

[DWOLATZKY 10]
Dwolatzky, B., Ng, Wan P., Chavez, Rafael S., *Process Improvement on a Regional Scale,* Keynote Panel of SEPG-NA, 2010.

[TYLPDE 09]
Thailand SPIN, *Accelerating CMMI Adoption with Personal Software Process (PSP)*, http://www.thailandspin.com/ThailandSPINCalendar/tabid/63/articleType/ArticleView/articleId/45/Default.aspx, 2009.

[SEI-PTNR]
SEI Partner Network, *http://www.sei.cmu.edu/partners/directory/*.

[SEI-MS 04]
SEI library, Microsoft's Pilot of TSP Yields Dramatic Results, *http://www.sei.cmu.edu/library/abstracts/news-at-sei/feature120042.cfm*, 2004.

[TSPSYMP]
Software Engineering Institute, *http://www.sei.cmu.edu/tspsymposium/2011/proceedings.cfm*, The Proceedings of Team Software Process Symposium.

[AKIYAMA 10]
Akiyama, Y., Katamine, K., Umeda, M., Hashimoto, M., Noma, T., *Improving Students' Ability in Developing High Quality Software using Personal Software Process (in Japanese),* SEC Journal, Vol. 6, No. 3, 2010.

[AizuUniversity 09]
Seminars on the next generation of software development process week, *http://web-ext.u-aizu.ac.jp/official/news/news179_j.html*, 2009.

[AizuUniversity 10]
Seminars on the disciplne for Quality management, *http://www.u-aizu.ac.jp/graduate/public-class/iclass/iclass05.html,* 2010.

[KAIYA 01]
Kaiya, H., Kojims A., Kaijiri, K., *Improving the education process in the university (in Japanese),* Proceedings of Software Symposium 2001, pp. 137-142, 2001.

8. BIOGRAPHY

Keiichi Katamine
Assistant Professor
Kyushu Institute of Technology

Keiichi Katamine is an assistant professor of the graduate school of computer science and systems engineering at Kyushu Institute of Technology. His research interests include business analysis, the specifications of information system, specification description languages, and their development environment based on software and knowledge modeling techniques. Moreover, he is interested in project and process management techniques to develop information systems. He received a Ph.D. in engineering from Kyushu Institute of Technology. He is a SEI authorized PSP instructor, and a Goldratt School certified Critical Chain Project Management (CCPM), and Simplified Drum-Buffer-Rope (S-DBR) trainer.

Masanobu Umeda
Associate Professor
Kyushu Institute of Technology

Masanobu Umeda is an associate professor of the graduate school of computer science and systems engineering at Kyushu Institute of Technology. His current research interests include knowledge-based systems and their development environment to support intellectual activities of domain experts in medical domain, product design and production domain, logistics domain, and so on. Moreover, process and management areas are in the research scope. Some research results are actually utilized in real-world applications. He also has many experiences with developing and distributing free/open software. He received Ph.D. in engineering from Kyushu Institute of Technology.

Masaaki Hashimoto
Professor Emeritus
Kyushu Institute of Technology

Masaaki Hashimoto has been studying the architecture of industrial knowledge and its analysis and modeling technique on the base of knowledge-based software engineering. He has been researching and developing a knowledge description language and automatic software generation technique. These researches and developments have been applied to actual business such as logistics, business systems, embedded systems, architectural design, ship building design, and so on. He also worked with an enterprise for more than 20 years.

Yoshihiro Akiyama
Founder & CEO
Next Process Institute Ltd.

Yoshihiro Akiyama is founder and CEO of Next Process Institute Ltd. Until March 2010, he was a professor at the graduate school of computer science and systems engineering and center for ICT Education, Kyushu Institute of Technology. He is a SEI-authorized PSP Instructor, SEI-Certified TSP Mentor Coach, and CMMI Instructor. Before KIT, he was a senior technical staff member at IBM, promoting process-based project management for IBM Asia Pacific. Since Joining IBM Japan in April 1974, he worked as systems engineer, research staff and manager for software engineering at IBM Tokyo Research Lab., and two times of oversea assignments to IBM US to develop a middleware product and to transfer software engineering technology. He earned a Ph.D. in particle physics in 1974. He received a Ph.D. in particle physics March 1974 from Tokyo Metropolitan University. He is a member of IEEE, PMI, IPSJ, and SPM.

Analysis of Design Defect Injection and Removal in PSP

Diego Vallespir, Universidad de la Republic
William Nichols, Software Engineering Institute

1.1 INTRODUCTION

A primary goal of software process improvement is to make software development more effective and efficient. One way of doing that is to understand the role of defects in the process and make informed decisions about avoiding defect creation or committing the effort necessary to find and fix the defects that escape development phases. Through examination of a large amount of data generated during Personal Software Process (PSP) classes, we can show how many defects are injected during design, what types of defects are injected, and how they are detected and removed in later development phases. We can use this information to teach developers how to define and improve their own processes, and thus make the product development more effective and efficient.

"The Personal Software Process (PSP) is a self-improvement process that helps you to control, manage, and improve the way you work" [Humphrey 05]. This process includes phases that you complete while building the software: plan, detailed design, detailed design review, code, code review, compile, unit test, and post mortem. For each phase, the engineer collects data on the time spent in the development phase and data about the defects injected and removed. The defect data include the defect type, the time to find and fix the defect, the phase in which the defect was injected, and the phase in which it was removed.

During the Personal Software Process course, the engineers build programs while progressively learning PSP planning, development, and process assessment practices. For the first exercise, the engineer starts with a simple, defined process (the baseline process, called PSP0); as the class progresses, new process steps and elements are added, from estimation and planning to code reviews, to design, and design review.

In this article, we present an analysis of defects injected during the design phase of the PSP programs 6, 7, and 8 (all developed using PSP2.1). In PSP2.1, students conceptualize program design prior to coding and record the design decisions using functional, logical, operational, and state templates. Students then perform a checklist-based personal review of the design to identify and remove design defects before beginning to write code.

In this analysis, we focused on defects injected during the design phase because these data had not been specifically studied before. Previous studies did not have all the defect data, such as defect types and individual defect times; they had only summaries. Our analysis of the complete data available from individual defect logs shows not only that the defects injected during design are the most expensive to remove in test but also these are easy to remove in the review phases. The difference is striking: it costs five times more to remove a defect in test than it does to remove that same defect during review.

To show this, we observed how defects injected during design escaped into each subsequent phase of the PSP and how the cost to remove was affected by defect type and phase. We describe the different defects types injected during design and how these defect types compare with respect to the "find and fix" time. From this analysis, we show that "function" defects are the most common design phase defect, that personal design review is an effective removal activity, and that finding and fixing design defects in review is substantially less expensive than removal in test.

Some other articles study software quality improvement using PSP [Paulk 10] [Wholin 98] [Rombach 08] [Paulk 06] [Hayes 97] [Ferguson 97]. In the context of PSP, quality is measured as defect density (defects/KLOC). Our study differs from these others in that we focus on design defects, consider the defect type, and do not consider defect density. Instead, we focus on the characteristics of the defects introduced in design. Our findings resulted from analyses of the defect types injected, how they proceeded through the process until they were found and removed, and cost of removal in subsequent development phases.

1.2 THE DATA SET

We used data from the eight program version of PSP for Engineers I and II taught between October 2005 and January 2010. These courses were taught by the Software Engineering Institute (SEI) at Carnegie Mellon University or by SEI partners, including a number of different instructors in multiple countries.

This study is limited to only consider the final three programs of the 2006 version of the PSP course (programs 6, 7, and 8). In these programs, the students apply the complete PSP process, using all process elements and techniques. Specifically, these exercises include the use of design templates and design reviews. Of course, not all of these techniques are necessarily applied well because the students are in a learning process.

We began with the 133 students who completed all programming exercises. From this we made several cuts to remove errors and questionable data and to select the data most likely to have comparable design and coding characteristics.

Because of data errors, we removed data from three students. Johnson and Disney reviewed the quality of the PSP data [Johnson 1999]. Their analysis showed that 5% of the data was incorrect; however, many or most of those errors in their data were due to calculations the students made. Because our data were collected with direct entry into a MS Access tool, which then performed all calculations automatically, the lower amount (2.3%) of data removed is lower than the percentage reported by Johnson and Disney but seems reasonable.

We next reduced the data set to separate programming languages with more common design and coding characteristics. As we analyze the design defects, it seems reasonable to consider only languages with similar characteristics that might affect code size, modularity, subroutine interfacing, and module logic. The students used a number of different program languages, as shown in Figure 1.

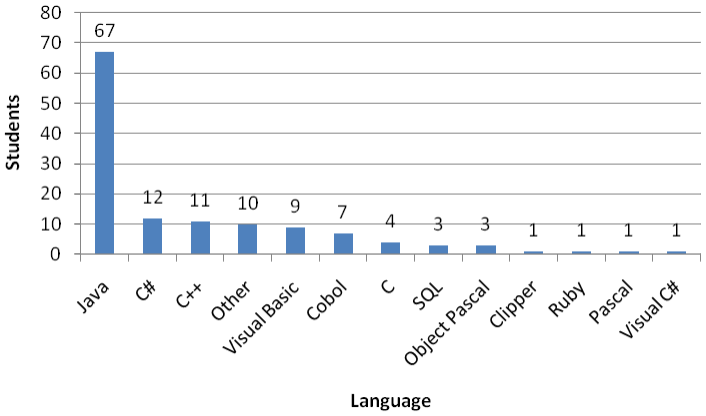


Figure 1: Quantity of students by program languages

The most common language used was Java. To increase the data set size, we decided to include the data generated by students who used Java, C#, C++, and C. This group of languages uses similar syntax, subprogram, and data constructs. For the simple programs produced in the PSP course, we judged that these were most likely to have similar modularization, interface, and data design considerations. This cut reduced our data to that generated by the programming efforts of 94 students.

Because our intent was to analyze defects injected in the design phase, we removed from consideration any data for which design defects were not recorded. From the 94 data sets remaining, two recorded no defects and 11 recorded no defects injected during design. Our data set for this analysis was, therefore, reduced to 92 engineers or 83 engineers depending upon the specific analysis performed. In the following sections we present the types of defects that were injected in the design phase, when the defects were removed, and the effort required to find and fix these defects.

1.3 WHERE THE DEFECTS ARE INJECTED

The first goal of our analysis was to better understand where defects were injected. We expected injections to be dominated by the design and code phases of course, because they are the construction phases in PSP. We began by explicitly documenting the phase injection percentages.

This analysis studied the defect injection and removal performance of individuals and the performance variation among them. We included the 92 engineers who recorded defects. We began by computing the phase data for each individual and then computed the following statistics of the distribution of individuals:

- an estimate of the mean percentage of defects injected by phase
- the 95% confidence interval for that mean (to characterize the standard error on the mean)
- the standard deviation of the distribution to characterize the spread among individuals

For each phase and for each individual, we calculated the percentage of defects injected in each PSP phase. The distribution statistics are shown in Table 1.

	DLD	DLDR	Code	CR	Comp	UT
Mean	46.4	0.4	52.4	0.3	0.03	0.5
Lower	40.8	0.2	46.7	0.0	0.0	0.2
Upper	52.0	0.7	58.1	0.7	0.09	0.9
Std. dev.	27.2	1.7	27.4	1.8	0.3	1.8

Table 1: Mean lower, upper confidence interval values and std. dev. of the % of defects injected by phase

The design and code phases have similar injection percentages both on average and in the spread. Their mean of the percentage of defects injected is near 50% with lower and upper confidence interval bounds between 40% and 58%. Both standard deviations are around 27% with. So, in the average of this population, roughly half of the defects were injected in the design phase and the other half of the defects were injected in the code phase. On average, the defect potential of these phases appears to be very similar. The standard deviation shows, however, that the variability between individuals is substantial. Nonetheless, as we expected, in the average almost 99% of the defects were injected in the design and code phases with only around 1% of the defects injected in the other phases.

The design review, code review, compile, and unit test phases also have similar average defect potentials. The average in all these cases is less than 0.5% and their standard deviations are small, the largest being 1.8% in code review and unit testing. This shows that during verification activities in PSP the percentage of defects injected is low but not zero. From time to time, developers inject defects while correcting other defects. We will study these secondary injections in a later study.

The variability between individuals and the similarity between the code and design phase is also presented in Figure 2. Note that the range in both phases is from 0% to 100% (all possible values). The 25th percentile is 26.34 for design and 35.78 for code, the median is 45.80 for design and 52.08 for code, and the 75th percentile is 64.22 for design and 71.56 for code.

Despite a high variability between individuals, this analysis shows that the great majority of defects are injected in the design and code phases. Slightly more defects are injected during code than during design, but the difference is not statistically significant. We could, therefore, focus on the defects injected in the design and code phases. In this article, we discuss only the defects injected in the design phase.

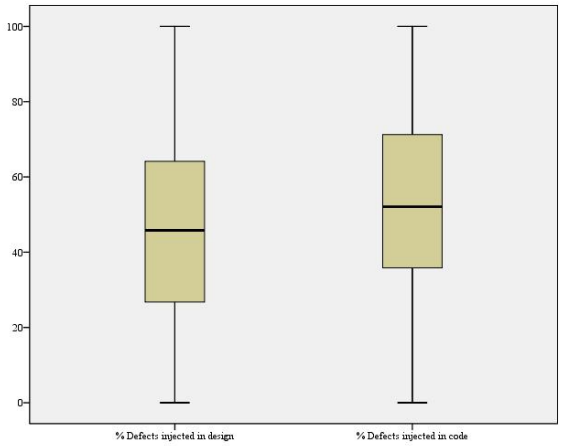


Figure 2: Percentage of defects injected by phase (box and whisker chart)

1.4 ANALYSIS OF DESIGN DEFECTS

From the 94 engineers in our data set there were 11 who recorded no injected defects during design. Our data set for analysis of the design defects was, therefore, reduced to 83 engineers. In the following sections we discuss the types of defects that are injected in the design phase, when those defects are removed, and the effort required to find and fix the defects.

1.4.1 Defect types Injected during Design

To improve the detection of design defects we first wanted to know which types of defects were injected during the design phase. Table 2 shows the mean of the percentage of the different defect types injected. It also presents the lower and upper bound of the 95% confidence interval for the mean (a measure of the standard error) and the standard deviation of the distribution.

We divided these defect types into three categories. The first is "almost not defects of this type." In this category we found system and build/package defects. It is clear that during the PSP course these types of defects were almost never injected. This may be due to the PSP course exercises rather than the PSP. Because the exercises are small, taking only a few hours, and contain few components, and make few external library references, build packages are usually quite simple. We expect to find more defects of these types in the Team Software Process in industrial scale projects. A second category is "few defects;" most of the other defect types (all except Function type) are in this category. The percentage of defects in this category ranged from 3.1% to 12.6%.

The last category, "many defects," includes only one type of defect: function. The great majority of defects injected during design were of the Function type. This type of defect was almost half of all the defects injected during Design. This is an observation familiar to PSP instructors, but not previously reported for a sizable data set.

The lower, upper, and standard deviation data show again the high variability between individuals. This can also be observed in Figure 3; the box and whisker chart shows many observations as outliers. Also, it can be seen that the function defect type goes from 0% to 100% and that the 25 percentile is 21% and the 75 percentile is 70%.

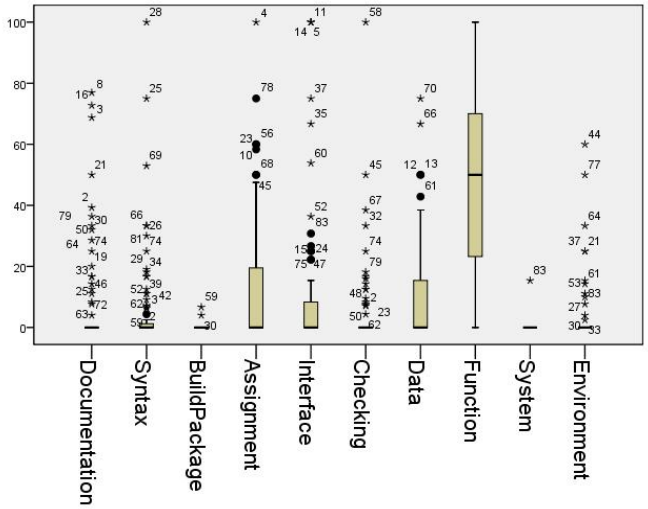


Figure 3: Box and whisker of the percentage of defects injected during design

1.4.2 When Are the Defects Injected During Design Removed?

Our analysis indicated the subsequent phases during which the design defects were removed. While our data set was larger than any previously studied, it remained too small for us to examine the removals based on defect type. Still, for each engineer who injected design defects, we identified the phases in which the engineers found the defects, then, for every phase, we determined the percentage of the defects that were found in that phase.

	Docs.	Syn.	Build	Assign.	Inter.	Check	Data	Func.	Syst.	Env.
Mean	6.9	6.0	0.1	12.6	10.0	4.6	9.8	46.6	0.2	3.1
Lower	3.3	2.5	0.0	8.2	5.1	1.6	6.3	39.7	0.0	0.9
Upper	10.5	9.5	0.3	17.0	15.0	7.6	13.3	53.5	0.6	5.3
Std. dev.	16.6	16.0	0.8	20.2	22.5	13.8	16.0	31.5	1.7	10.1

Table 2: Percentage of defect types injected during design

Table 3 shows the mean (with 95% confidence interval) and standard deviation for the different phases. The 95% confidence interval is bounded by 45.8% and 61.0%. As previously shown, the standard deviation was high, indicating the high variability between individuals. From this we learned that approximately 50% of the defects injected during Design were found in the detailed level design review (DLDR) phase.

	DLDR	Code	CR	Comp	UT
Mean	53.4	9.6	8.9	2.5	25.7
Lower	45.8	5.7	5.2	0.0	19.3
Upper	61.0	13.4	12.5	5.2	32.0
Std. dev.	34.8	17.5	16.7	12.3	29.2

Table 3: Phases where are founded the design defects (percentage)

Figure 4 shows the box and whisker charts displaying the percentage of defects found in the different phases and the histogram for the defects founds in DLDR. Figure 4 also shows the high variability between individuals in the percentage of defects found during DLDR and UT.

Code and code review have a similar percentage of defects that were injected during design. Approximately 10% of the defects were found and removed in each of those phases. Approximately 2.5% of the design defects were found during the compile phase; it is likely that the defects found in compile were pseudo-code defects. And finally, around 25% of the defects were found in unit test (UT). This means that in the PSP accounting, one of every four defects injected during design escapes all phases prior to UT. We know, of course, that not all the defects that escape into UT are found in UT. UT will not have a 100% yield; therefore the percentage of defects found in each of these phases is smaller than reported while the actual percentage of escapes into UT is a lower limit. An estimate or measurement of the UT yield will be necessary to revise these phase estimates.

1.4.3 Cost to Remove the Defects Injected in Design

What is the cost and variation in cost (in minutes) to find and fix the defects that are injected during design? First, we analyze the differences in cost segmented by the removal phase. Second, we study the differences in cost segmented by defect type.

It would also be interesting to segment and analyze both the removal phase and the defect type jointly. Unfortunately, because of limited sample size after a two dimensional segmentation, we cannot perform that analysis with statistical significance.

1.4.3.1 Phase Removal Cost

What is the cost, in each removal phase, to find and fix a defect injected in design? Design defects can be removed in the detailed level design review (DLDR), code, code review (CR), compile, and unit test (UT) phases. For each engineer, we calculated the average task time of removing a design defect in each of the different phases. Because some engineers did not remove design defects in one or more phases, our sample size varied by phase. We had data from 67 engineers for DLDR, 29 each for code and CR, six for compile, and 55 for UT. We excluded the cost of finding design defects in the Comp phase because we had insufficient data for that phase.

Table 4 shows the mean, lower and upper 95% confidence interval, and the standard deviation for the find and fix time (in minutes) for design defects in each of the studied phases. We might have expected an increased cost in each phase but this is not what the data showed.

	DLDR	CODE	CR	UT
Mean	5.3	5.1	4.2	23.0
Lower	3.7	2.5	2.6	11.6
Upper	6.9	7.6	5.7	34.3
Std. dev.	6.6	6.7	4.1	42.0

Table 4: Cost of find and fix defects injected in design segmented by phase removed

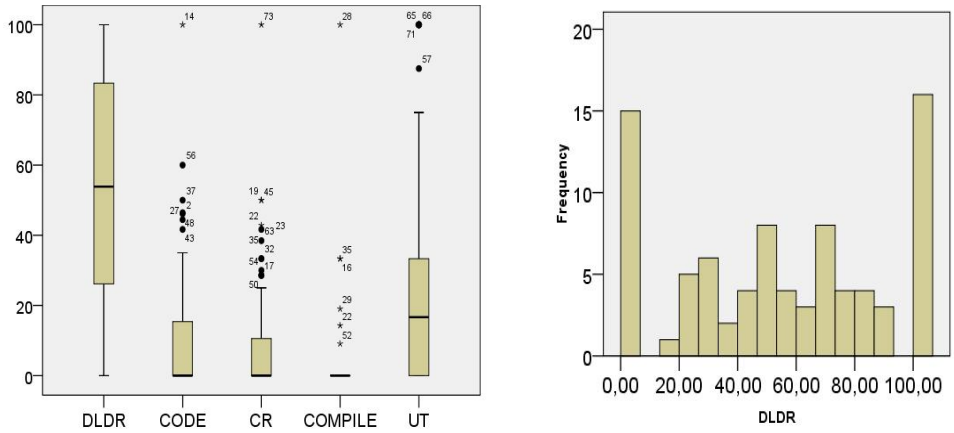


Figure 4: In which phase are the design defects found? – Variability between individuals

Rather, **the "find and fix" cost remained almost constant during DLDR, code, and CR**; in fact, the cost decreased a little in these phases, though the differences were not statistically significant. Further analysis will be needed to determine which of the defect finds in code and code review escaped through design and an effective (as opposed to ineffective) design review. Regardless, any defect discovered in these phases is essentially found by an inspection process where the "fix" time is short because the root cause has been identified.

We are not stating here that the cost of finding and fixing a design defect during DLDR, code, and CR is necessarily the same. We are stating that using PSP, the design defects that are removed during DLDR cost approximately the same as removing the ones that escape from design into code and those that escape from design into CR.

As we expected**, the average cost of finding a design defect during UT is much higher than in the other phases** by almost a factor of 5.

We also found a high variability among individual engineers. This variability can be seen in the box and whisker chart in Figure 5. We tested for normal distribution after log transformation of find and fix times for DLDR, code, CR, and UT and all are consistent ($p > 0.05$ using a Kolmorov-Smirnov and Shapiro-Wilk test) with a log-normal distribution. This test is primarily useful to verify that we can apply regression to the transformed data; however, understanding the distribution also helped to characterize the asymmetry and long tailed nature of the variation. That is, the log-normality affirmed our intuition that some defects found in test required far more than the average effort to fix, making test time highly variable. We also observed that the both the mean and variation of rework cost in the DLDR, code, and CR phases were significantly lower than UT in both the statistical sense and the practical sense.

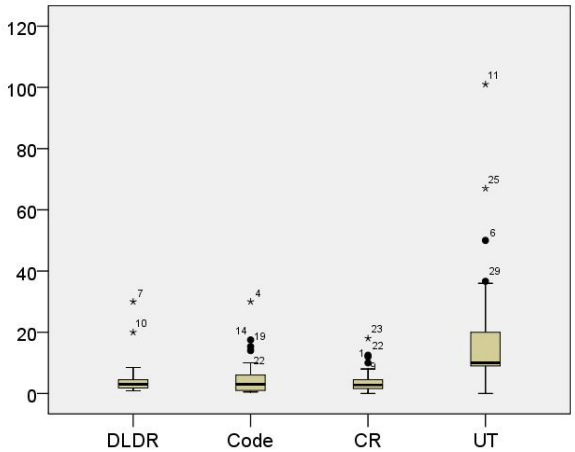


Figure 5: Box and whisker of the cost of find and fix a design defect segmented by phase removed

1.4.3.2 Defect Removal by Type

What is the find and fix cost, per defect type, of defects injected during detailed design? As we mentioned before, we had few build/package and system defects injected during design. As a result, we didn't have enough data for a statistically significant analysis of the cost of removing these two types of defects. However, we were able to analyze the remaining defect types.

Table 5 presents the mean, lower, and upper 95% confidence interval and the standard deviation for the find and fix cost of design defects, segmented by type. The cost, in minutes, for "find and fix" fell into three groups:

- a group that has a mean near 5 minutes: Documentation, Syntax, Interface, Checking
- a group, composed only of Assignment defects, that has a mean near 7 minutes
- a group that has a mean near 10 minutes: Data, Functions, Environment

The confidence interval of the second group is sufficiently wide that it is not clearly distinct from either of the other two groups, falling more or less in between. More data would help to clarify this grouping. We want to emphasize that the third group, including data, functions, and environment, takes twice the time to find and fix the defects than the documentation, syntax, interface and checking types of the first group.

	Docs.	Syn.	Assign.	Inter.	Check	Data	Func.	Env.
Mean	5.6	4.3	7.3	5.4	4.9	11.0	9.3	10.5
Lower	3.6	1.8	1.9	2.5	2.2	2.2	6.9	3.0
Upper	7.6	6.7	12.7	8.2	7.5	19.8	11.7	17.9
Std. dev.	4.1	3.7	16.3	7.3	5.2	25.6	10.1	11.7

Table 5: Cost of find and fix defects injected in design discriminated by defect type

As in the other cases, the variation among individual developers was high. This can be seen using the standard deviation, as well as the box and whisker chart that is presented in Figure 6.

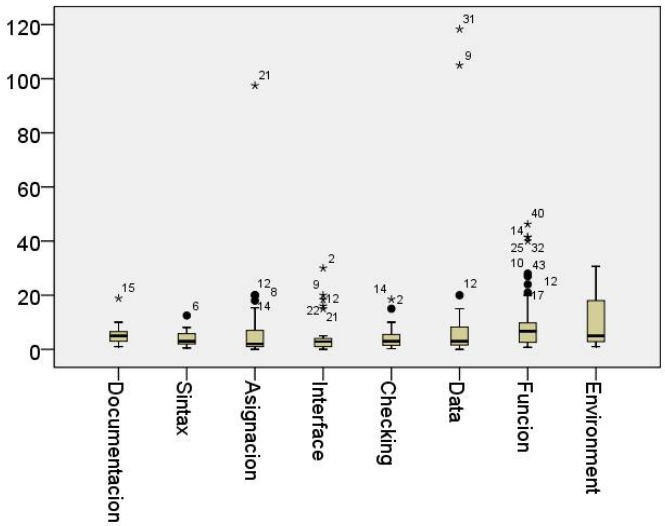


Figure 6: Box and whisker of the cost of find and fix a defect segmented by defect type

1.5 CONCLUSIONS AND FUTURE WORK

In this analysis, we considered the work of 92 software engineers who, during PSP course work, developed programs in the Java, C, C#, or C++ programming languages. In each of our analyses, we observe that there exists a high variation in range of performance among individuals; we show this variability using standard deviation and box and whisker charts to display the median, quartiles, and range. After considering this variation, we focused our analysis on the defects injected during design. Our analysis showed that most common design defects (46%) are of type function. This type belongs to the group of the most costly defects to find and fix. Data and environment defect types are in the same cost group category as Function.

In addition, the analysis showed that build/package and systems defects were seldom injected in the design phase. We interpreted this as a consequence of the small programs developed during the course, rather than as a characteristic of PSP as a development discipline.

In the final three PSP course exercises, defects were injected roughly equally in the design and code phases; that is, nearly half of the defects were injected in design. Half of the design defects were found early through appraisal during the detailed design review (DLDR). However, around 25% were discovered during unit test, where defect find and fix is almost five times more expensive in time.

While this analysis provided insights into the injection and removal profile of design defects with greater specificity than previously possible, a larger data set would allow us to consider more detail, such as the costs of defects discriminated by defect type in addition to removal phase. A more complete analysis, including a study about the defects injected during the code phase, may enable us to analyze improvement opportunities to achieve better process yields.

In future analysis, we will examine the relationship between design and code activities and the defects found in the downstream phases. In particular, we want to determine how variation in design and design review affects defect leakage into these later phases. During a related analysis, we will examine the effects of design and design review on secondary defects injected in the unit test phase.

1.6 REFERENCES/BIBLIOGRAPHY

[Ferguson 97]
Ferguson, Pat; Humphrey, Watts S.; Khajenoori, Soheil; Macke, Susan; Matvya, Annette. *Results of Applying the Personal Software Process,* Computer, vol. 30, no. 5, pp. 24—31, 1997.

[Hayes 97]
Hayes, Will; Over, James. *The Personal Software Process: An Empirical Study of the Impact of PSP on Individual Engineers,* Technical Report, Carnegie Mellon University, Software Engineering Institute, no. 97-001, 1997.

[Humphrey 05]
Humphrey, Watts S. *PSP A Self-Improvement Process for Software Engineers,* Addison-Wesley, 2005.

[Johnson 99]
Johnson, Philip M.; Disney, Anne M. *A Critical Analysis of PSP Data Quality: Results from a Case Study,* Empirical Software Engineering, vol. 4, no. 4, 317—349, 1999.

[Paulk 06]
Paulk, Mark C. *Factors Affecting Personal Software Quality,* CrossTalk: The Journal of Defense Software Engineering, vol. 19, no. 3, pp. 9—13, 2006

[Paulk 10]
Paulk, Mark C. *The Impact of Process Discipline on Personal Software Quality and Productivity,* Software Quality Professional, vol. 12, no. 2, pp. 15—19, 2010.

[Rombach 08]
Rombach, Dieter; Munch, Jurgen; Ocampo, Alexis; Humphrey, Watts S.; Burton, Dan. *Teaching disciplined software development,* The Journal of Systems and Software, vol. 81, no. 5, pp. 747—763, 2008.

[Wholin 98]
Wohlin, C.; Wesslen, A. *Understanding software defect detection in the Personal Software Process,* Proceedings of the Ninth International Symposium on Software Reliability Engineering, pp. 49—58, 1998.

2. BIOGRAPHY

Diego Vallespir

Assistant Professor
Universidad de la República

Diego Vallespir is an Assistant Professor at the Engineering School at the Universidad de la República, Director of the Informatics Professional Postgraduate Center at the same school, Director of the Software Engineering Research Group (GrIS) at the Universidad de la República and member of the Organization Committee of the Software and Systems Process Improvement Network in Uruguay (SPIN Uruguay).

Vallespir holds an Engineer title on Computer Science from Universidad de la República and a Master Science title in Computer Science from the same University. He has several articles published in international conferences. His main research topics are empirical software engineering, software process and software testing. He is currently finishing his PhD Thesis.

Bill Nichols

Senior Member of the Technical Staff
Software Engineering Institute

Bill Nichols joined the Software Engineering Institute (SEI) in 2006 as a senior member of the technical staff and serves as a PSP instructor and TSP coach with the Team Software Process (TSP) Program. Prior to joining the SEI, Nichols lead a software development team at the Bettis Laboratory near Pittsburgh, Pennsylvania, where he had been developing and maintaining nuclear engineering and scientific software for 14 years. His publications include the interaction patterns on software development teams, design and performance of a physics data acquisition system, analysis and results from a particle physics experiment, and algorithm development for use in neutron diffusion programs. He has a doctorate in physics from Carnegie Mellon University.

Thank You

Thanks to the TSP Symposium 2011 Program Committee
Timothy Chick
Software Engineering Institute, TSP Symposium 2011 Technical Chair
Senior Member of Technical Staff, SEI Software Engineering Process Management Program

Lana Cagle, U.S. Navy
Barry Dwolatzky, Johannesburg Centre for Software Engineering
João Pascoal Faria, Faculdade de Engenharia da Universidade do Porto
Bill Nichols, Software Engineering Institute
David Saint-Amand, U.S. Navy
Rafael Salazar, Tec de Monterrey
Kathy Smith, HP Enterprise Services
David Webb, U.S. Air Force

Thanks to the TSP Symposium 2011 Planning Committee
Bob Rosenstein, SEI Conferences, Events, and Trade Shows Manager
Michele Falce, SEPM Technical Event Coordinator
Ruth Gregg, TSP Symposium Lead Event Planner
Brittney Osikowicz, Public Relations Coordinator

Thanks to the SEI Partner Network Staff
Lisa Masciantonio, SEI Partner Network Manager
Tracey Kelly
Kay Vinay

Thanks to the SEI Communication Design, Web Communication, and Customer Relations Staff
Karen Balistreri
Jeff Balmert
Deen Blash
Jeff Federoff
Maureen Fechik
David Gregg
Shane McGraw
Bill McSteen
Anna Mosesso
Melissa Neely
Daniel Pipitone
Trish Schreiber
Lizann Stelmach
Cat Zaccardi

Thank you to the Atlanta SPIN for its warm welcome and hospitality.

Be sure to bookmark www.sei.cmu.edu/tspsymposium to get this year's proceedings and for information on future TSP events hosted by the SEI.

Upcoming SEI Conferences and Events

Visit www.sei.cmu.edu/events/ to get the latest information on upcoming conferences and events.

October 24-26, 2011
CMMI Workshop 2011
Minneapolis, Minnesota

November 2-3, 2011
SEPG Latin America 2011
Lima, Peru

March 12-15, 2012
SEPG North America 2012
Reaching New Levels of Excellence
Albuquerque, New Mexico

June 5-7, 2012
SEPG Europe 2012
¡A Passion for Process!
Madrid, Spain

August 2012
SEPG Asia-Pacific 2012
Melbourne, Australia