Carnegie Mellon University
Software Engineering Institute

Software Excellence Through the Agile High Velocity Development℠ Process

Barti K. Perini, Vice President, Software Process Improvement, *ISHPI*
Stephen M. Shook, Vice President, Software Engineering & Quality, *ISHPI*

July 2023

TECHNICAL REPORT
CMU/SEI-2023-TR-002
DOI: 10.58012/R53E-3280

Ishpi Information Technologies, Inc. (DBA *ISHPI*) – Advanced Information Services Division

http://www.sei.cmu.edu

Design: REV-03.18.2016.0 | Template: 01.04.2023

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8702-15-D-0002 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

This report was prepared for the SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100

DM23-0663

Table of Contents

List of Figures

List of Tables

Acknowledgments

The authors would like to acknowledge Watts Humphrey, whose dedication to software quality has been an inspiration to us. The methods he developed while at the Software Engineering Institute (SEI) provided the foundation for our improvements.

This journey has only been possible because of the wholehearted support of Earl Bowers and the rest of the *ISHPI* management and corporate team. Their leadership sustains our culture of commitment to consistently deliver secure and high-quality results for our customers.

ISHPI will continue to advance because of the principles put in place by the leadership of Girish Seshagiri and the contributions of so many current and former Advanced Information Services (AIS) Division managers and team members over the years.

While leading us through six high-maturity Capability Maturity Model Integration (CMMI) appraisals over the past 15 years, CMMI Lead Appraisers Edward Weller and Shane Atkinson provided invaluable insights and industry perspective to achieve our business objectives.

We would like to express our appreciation to the Software Excellence Alliance (SEA) for providing a supportive community and inspiring forum for sharing ideas and techniques for achieving high-quality, customer-pleasing results.

Finally, we thank Julia Mullaney, Michele Falce, and Edward Desautels for their guidance and support in writing this technical report and for their proactive coordination throughout the Watts S. Humphrey Software Quality Award process.

Executive Summary

Ishpi Information Technologies, Inc. (DBA *ISHPI*) is an award-winning Service-Disabled, Veteran-Owned Small Business (SDVOSB), specializing in information technology (IT) services, cybersecurity services, and secure software development. To date, *ISHPI* has performed U.S. government contracting work centered on the delivery of high-quality IT services and support to the Department of Defense (DoD), Department of Homeland Security (DHS), and other agencies of the U.S. federal government for more than 16 years.

We skillfully leverage our management expertise and ISO 9001:2015-certified quality management, ISO/IEC 20000-1:2018 IT service management, and ISO/IEC 27001:2013 information security management systems and use our Capability Maturity Model Integration (CMMI) for Development (CMMI-DEV v2.0) Maturity Level (ML) 5 and CMMI for Services (CMMI-SVC v2.0) Maturity Level 3 practices to perform a variety of services for the federal government, including software engineering, cybersecurity, IT, and engineering and technical services. *ISHPI* has demonstrated the capability to perform all aspects of software engineering and development, including the development and sustainment of business solutions; the use of enterprise architectures; and the performance of requirements management, software engineering, systems engineering, software test management, and systems/server administration.

In 2014, *ISHPI* acquired Advanced Information Services Inc., a globally recognized leader in software development quality operating at CMMI ML 5 and winner of the 2013 Government Information Security Leadership Award (GISLA) and the IEEE Computer Society's 1999 Software Process Achievement (SPA) Award. This organization, now the *ISHPI* AIS Division, specializes in secure software development, modernization, maintenance, enhancement, and sustainment, and it has managed more than 250 software projects. The AIS Division is the functional area within *ISHPI* that exercises management control of the project and its processes. Overall, *ISHPI* consists of 78 employees, including 13 individuals from the AIS Division.

In 1999, AIS received the SPA award. From 1999 through 2005, AIS continued its process improvement efforts using the Capability Maturity Model (CMM) framework as an enabler to achieve business objectives. At about the same time, a majority of the business contracts for AIS changed from "time and material" type contracts to "firm fixed price." The AIS Division started working on federal contracts that required handling sensitive information, such as personally identifiable information (PII). Consequently, it had to adapt to using a secure software lifecycle to consistently deliver software with zero cybersecurity vulnerabilities in the code. In addition, the AIS Division offered customers an unprecedented lifetime warranty against software defects as a strategic long-term competitive differentiator [Shull 2013]. To address these business needs, the AIS Division needed to continue to improve effort, cost, and schedule estimates and focus on quality. As a result, AIS Division senior management established the following strategic goals for the AIS Division:

- Continue to grow and be profitable by making quality and security the highest priority.
- Consistently deliver defect-free software, on time every time, and eliminate cost overruns.
- Make commitments based on plan, capability, and historical data.

- Provide value to customers through organizational capability, team productivity, and individual discipline.
- Produce self-managed teams mentored by a qualified team coach (Section 1.5.4) and the Software Center of Excellence (SCOE) (Section 1.2).

We realized that the continuous process improvement goal to attain CMMI ML 5 must enable secure software development and disciplined capability to support guaranteed quality and firm-fixed-price deliveries with predictable effort and on schedule to be profitable. Additionally, the process must support business objectives, such as zero vulnerabilities in the source code and significantly shortened test cycles, to achieve high productivity for projects to be profitable. Furthermore, the process had to address agile iterative incremental development to enable close collaboration between the project team and the customer to get it right the first time and deliver high quality without accumulating technical debt.

Disciplined incorporation of these elements resulted in the Agile High Velocity Development℠ (HVD) Process (Section 1)—a significant innovation for the organization.

The *ISHPI* AIS Division's HVD process is based on scalable CMMI ML 5 processes, tailored to meet team size, customer standards, and project requirements to ensure the lowest lifecycle cost and greatest customer satisfaction. The high-maturity processes supported by team coach and SCOE enable individuals and teams to execute at their highest possible level of performance. The AIS Division's software development process was appraised at CMMI ML 5 by external appraisers in 2007, 2010, 2014, 2019, and 2022. The HVD processes, best practices, tools, and templates, containing over two decades of improvements, are available on the *ISHPI* process asset library, iPAL. In addition to appraising the AIS Division for CMMI-DEV for ML 5, the parent organization *ISHPI* has undergone a CMMI-SVC appraisal and achieved ML 3.

Abstract

The Advanced Information Services Division of Ishpi Information Technologies, Inc. (DBA *ISHPI*) performs all aspects of the software development lifecycle using its High Velocity Development℠ (HVD) process. We have studied many methods and frameworks (including Personal Software Process, Team Software Process, CMMI for Development, Scrum, Kanban, CMMI for Services, ISO 9001 [Quality Management], ISO 20000-1 [Information Technology Service Management], ISO 27001 [Information Security Management Systems], Cybersecurity Maturity Model Certification, and more), adapted them, combined them, and made them our own. The result is an innovative, cohesive process that works for us—our agile HVD process. We have shown that diverse inputs need not be contradictory choices, but instead complementary building blocks. By evolving, implementing, and utilizing the HVD practices, AIS Division teams have achieved significant improvement in productivity and performance. *ISHPI*'s customers have benefited from shorter schedules, lower costs for development due to minimal rework costs, lower costs for maintenance, and an overall positive experience during each project.

1 Agile High Velocity Development℠ (HVD) Process

HVD is *ISHPI*'s set of agile software practices for development, maintenance, and modernization that have enabled us to achieve results that are far superior to industry averages for schedule, cost, and quality. HVD uniquely blends the responsiveness of Agile with the discipline of CMMI ML 5 practices. Unlike the waterfall method of software development, HVD is iterative, flexible, and proactively accommodates change. The practices are lean, inherently flexible, and highly tailorable to meet the team size, customer standards, delivery schedules, and project requirements of individual contracts. These practices have been repeatedly appraised at CMMI ML 5 in every appraisal during the last 17 years. HVD focuses on quality to produce secure software with the lowest lifecycle cost, lowest risk, and greatest customer satisfaction. HVD is tool and technology agnostic, and it is more than a methodology (Figure 1). Furthermore, the results we have achieved using HVD are far superior to industry averages for schedule, cost, and quality (Table 8) [Standish Group 2013].

ISHPI AIS performs software development by tailoring the HVD process to the organization- and project-specific requirements of each project. It is not prescriptive, not "one-size-fits-all," not bureaucratic, and it is not *just* a methodology. Rather, it is a toolbox of policies, processes, procedures, standards, templates, instructions, guidelines, checklists, statistical techniques, tools, and utilities developed to jump-start and support project teams.

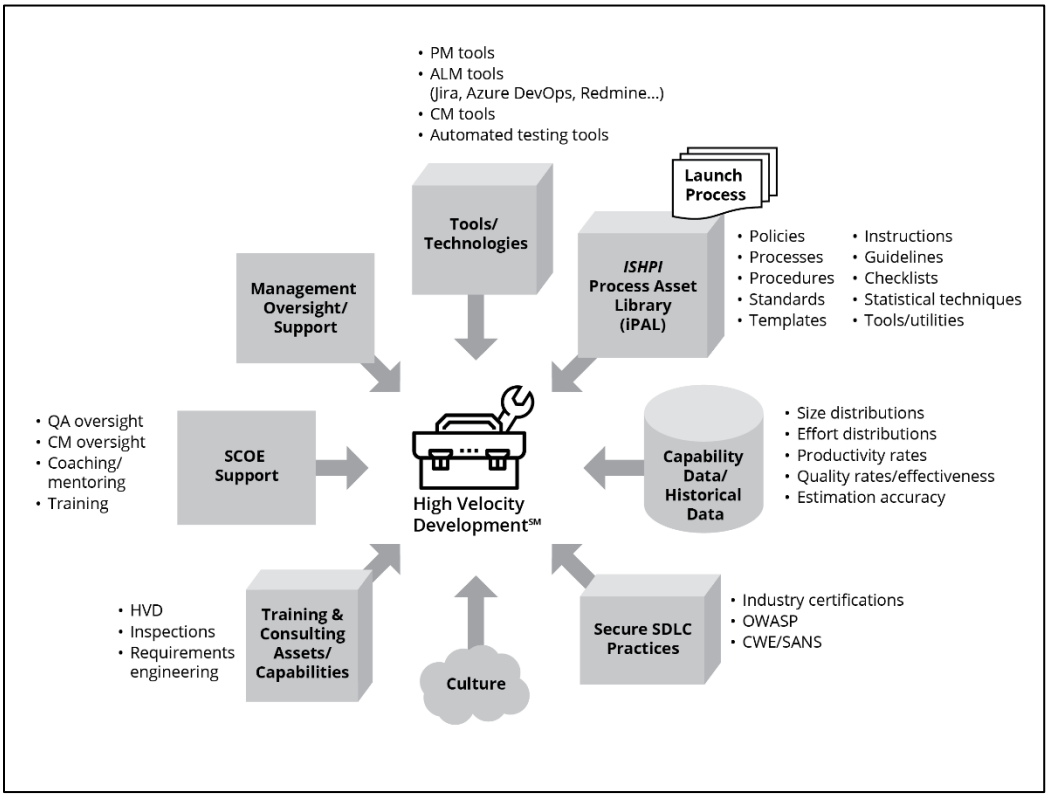

Figure 1: High Velocity Development℠ (HVD)—More than a "Methodology"

The HVD method emphasizes quality and excellence by using

- Personal Software Process (PSP℠) at the individual level
- Team Software Process (TSP℠) at the team level
- disciplined data at the organization level
- disciplined-but-agile practices based on CMMI for Development and CMMI for Services

We initially designed the HVD process to support business objectives, such as zero vulnerabilities in the source code and significantly shorter test cycles, for projects to be profitable. Over time, the process had to also address agile iterative incremental development to enable close collaboration between the project team and the customer to get it right the first time and deliver high quality without accumulating technical debt. Disciplined incorporation of these elements resulted in today's HVD Process.

The attributes of HVD are

- self-managed teams, supported by a qualified team coach
- team members trained in disciplined software engineering principles
- development strategy aligned with business goals
- frequent testable increment deliveries
- lowest overall cost of quality and highest value
- predictable cost, schedule, and quality performance
- managed using data
- continuous improvement

1.1 High Velocity Development℠ Execution

1.1.1 Project Planning

Project teams using HVD use the Team Launch Process at the beginning of the project and at the beginning of each increment and/or major release to set project goals, establish the overall project schedule, work breakdown structure (WBS), and milestones. The objective of the launch process is to develop team goals, roles, development strategy, process, preliminary development plans, risks, performance objectives, and measurement criteria (Figure 3) [Davis 2003, McAndrews 2000].

The Launch Process scripts and standards are in *ISHPI's* web-based process asset library, iPAL, for ready use and are summarized under Team Launch Process in Figure 2. With input from *ISHPI* internal management, customer stakeholders, and the project roadmap, the launch engages the whole *ISHPI* team to establish the goals and scope of the release and estimate the level of effort, timelines, activities, deliverables, and other work products that are needed to achieve those goals and scope. At the conclusion of the team launch, the team presents its plan and any alternates to customer stakeholders. Upon approval by the customer stakeholder, the *ISHPI* Project Manager makes updates to the Project Management Plan, WBS, Release Roadmap, and overall Project Plan. The milestone dates for the release are also established.

Release planning using the Team Launch Process occurs at regular intervals in conjunction with releases, as identified in the WBS. "Re-plans" also occur, in consultation and concurrence with customer stakeholders, when objectives or scope of the release or plan as a whole change or when there are variances causing significant deviations from the project schedule that justify a re-plan.

When executing the project, the project teams manage the goals and performance to meet customer objectives. The team understands and uses the capabilities of *organization*, *project*, and *individual* to meet their objectives. They perform causal analyses and retrospectives to continuously improve individual, project, and organization processes (Figure 2).

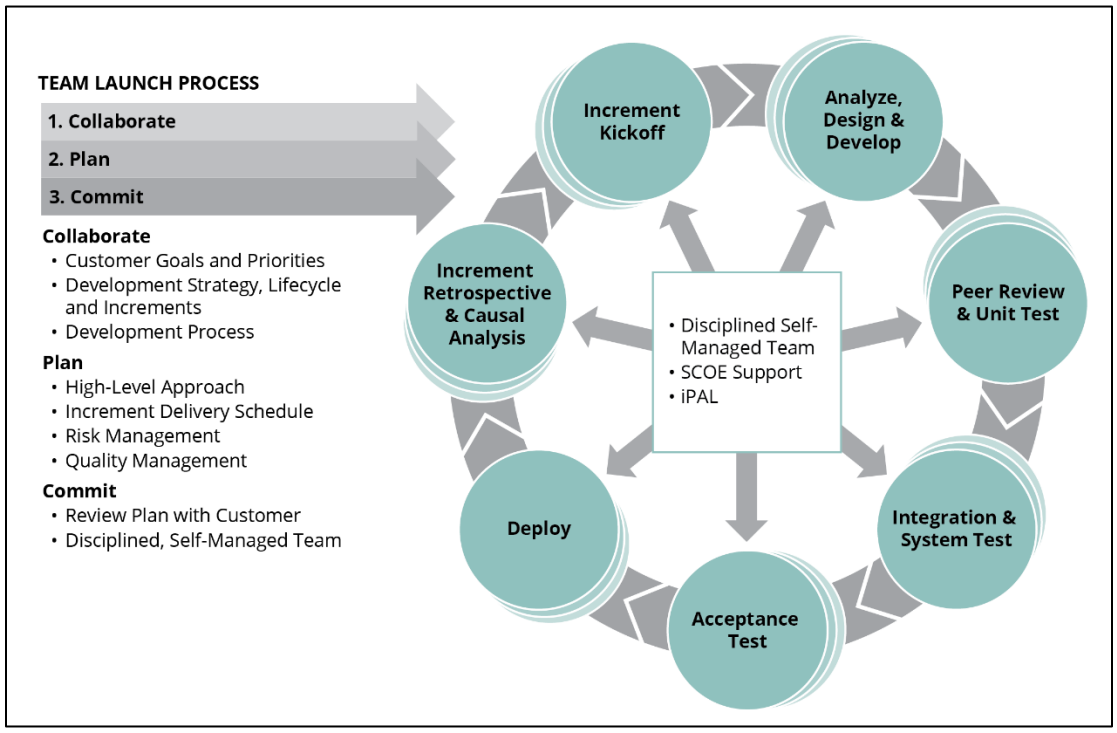

Figure 2: High Velocity Development℠ (HVD)

The objective of the launch process is to develop team goals, roles, development strategy, process, preliminary development plans, risks, performance objectives, and measurement criteria (Figure 3).

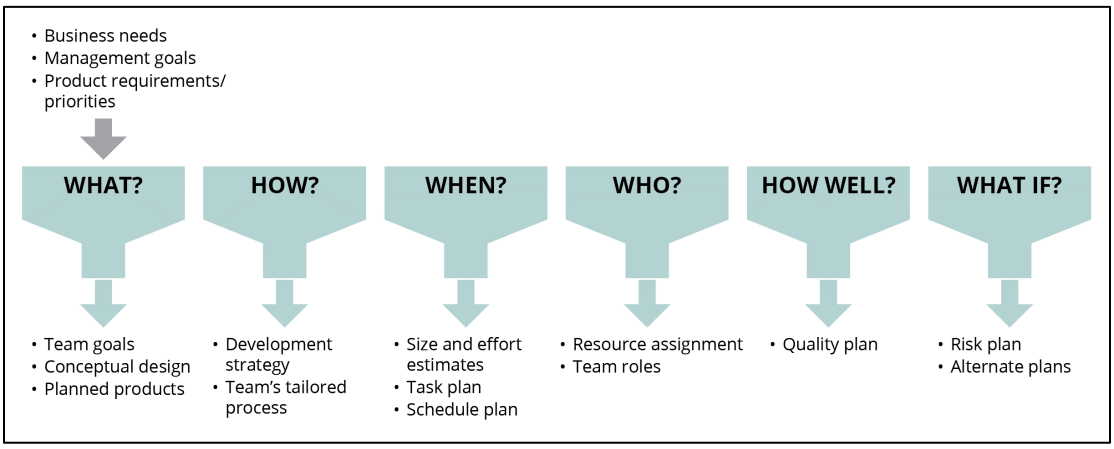

Figure 3: Project Launch Objectives

1.1.2 Project Execution and Implementation

Applying agile principles, the team implements the requirements in iterations. For each iteration, a kickoff meeting occurs, in which the team plans the increment and establishes specific technical details and content. The team elaborates the plan from the launch while staying within the constraints of the established overall project schedule. Within each iteration, individual developers take ownership of one or more components to produce the detailed design and code, and execute unit tests. After one or more iterations, the team performs a system/integration test on the functionality developed to that point. After the test, the team conducts an internal audit to ensure that deliverables are of high quality and are ready for subsequent customer testing. Each such group represents functionality that could be released for acceptance or demonstrated. The team participates in and supports all customer-required tests and gate reviews prior to production rollout (Figure 2). The AIS Division Software Center of Excellence (SCOE) assists project teams by providing a team coach and supporting configuration management (CM) and quality assurance (QA) activities. Additionally, the SCOE representative supporting the team reviews the project plan before it is presented to the customer for approval to ensure that quality and security-related tasks such as personal reviews, peer reviews, unit tests, system tests, and scans are included in the plan and not performed as an afterthought.

1.1.3 Performance Metrics

The project team identifies performance metrics that they will track and report as a driver to ensure that the performance standards and customer-specified acceptable quality levels will ultimately be met. The team establishes these metrics with the first project launch as part of the goals meeting. These metrics guide the team and provide in-progress evaluation that the performance standards, along with other project goals, are on target to be met. Our Monthly Status Reports include reports on performance metrics. An example of a team's identified performance metrics and its tracking is given in Table 1.

Table 1: Example of a Team's Performance Metrics

#	Performance Metric	Measure	Target	Tracking Notes	Example Status As of 5/31/2023
1	On-time releases	Ratio of the number of releases delivered to User Acceptance Test (UAT) on or ahead of schedule to the total number of releases	1.00	Measured after each release deployed to UAT	1.00
2	Complete functionality	% of functionality, as measured by completed user stories, delivered as compared to the number of committed stories	>= 100%	Measured after each increment	100%
				# delivered	41
				Total # committed	41
3	Completion of deliverables	% of committed deliverables delivered on time (or ahead of schedule)	100%	Measured each month	98.4%
				Cumulative # delivered on or ahead of schedule	124
				Total # delivered	126
4	Schedule deviation for the current project plan	% deviation on percent of work complete = (total # of projected days in current project planned – baselined # of days) / baselined # of days	< = 5%	Measured each month	-11.9%
5	Responsiveness to requests	% of inquiries (non-Tier 3 support requests) that "Respond to staff and acknowledges inquiry within one business day"	100%	Measured each month	100%
6	Responsiveness to Tier-3 requests	% of Tier 3 support requests that "Respond within 15 minutes of notification"	100%	Measured each month	100%
7	Deliver high-quality releases	% of user stories requiring no rework after delivery for UAT	Cumulative >= 95%	Measured after each build deployed to Test	100%
				Total user stories delivered	41
8	Deliver releases requiring no rework after UAT or after deployment to production	Post-UAT defects injected this contract year and discovered during UAT/production use	<= 1 for the year	Measured with each release	0
9	Deliver high-quality deliverables with excellent written communication	% of deliverables requiring no rework and without negative government comment after final delivery	Cumulative >= 95%	Measured monthly	100%
				# without negative comment	141
				Total # delivered	141

1.1.4 Project Tracking and Reporting

During project execution, the project teams manage the goals and performance to meet customer objectives. The team understands and uses the capabilities of organization, project, and individual to meet their objectives. They perform causal analyses and retrospectives to continuously improve individual, project, and organization processes (Figure 2).

ISHPI tracks project status and discusses progress, issues, risks, and interdependencies during the meetings, reports, and reviews identified below.

Team Status Meeting: Team status is reported at the following meetings:

- Daily Stand-Up Meeting (Daily Scrum): Team members meet daily to discuss the progress made since the last meeting, the current work plan, and if there are any roadblocks.
- Bi-Weekly Meeting: Team members meet once every two weeks to review in depth the status of the project, goal performance, issues, risks, quality, and any intergroup coordination needs.

Development Manager Meeting: The Project Manager meets with the *ISHPI* Program Manager or the Development Manager periodically to review project status, goals, risks, status of critical problems, staffing needs, and training needs.

Customer Status Reporting: Customer status is reported through the following channels:

- Weekly Customer Status Meeting: The Project Manager meets with customer stakeholders to discuss project progress, critical problems, any intergroup coordination needs. A written status report accompanies the meeting.
- Monthly Status Report (MSR): The Project Manager provides a status report and meets with customer stakeholders to review project status, progress, performance metrics, critical problems, status of support requests, and any other items of importance.
- Annual Program Review: The Project Manager meets with customer stakeholders annually to review project status, milestones and achievements, status of in-progress/not-started activities, and any other items of importance.
- Increment Review: The team meets with customer stakeholders to review its performance during the increment/release and get customer feedback.

1.1.5 Quality Management

The team performs quality management activities throughout performance of the project as specified in the team's Quality Assurance Plan (QAP). The plan addresses steps that the team will take to ensure that the quality of the work products to be developed complies with our defined processes. The team places appropriate verification and validation processes under statistical process control, and the details are documented in the QAP.

The QA representative from our SCOE objectively oversees the team's quality practices. The QA representative performs audits according to the defined process to ensure that the team follows practices defined by the QAP and that the associated verification and validation processes such as personal reviews, peer reviews, and tests are effective and result in high-quality deliverables. The QA representative and the project team monitor appropriate statistical process data to ensure

performance is within acceptable limits, process improvements are effective, and quality capability is maintained.

1.2 The Role of the *ISHPI* Software Center of Excellence (SCOE)

ISHPI has a SCOE that provides Software Engineering Process Group (SEPG) leadership, QA, CM, coaching led by qualified team coaches, and training support to project teams to ensure standards compliance and CMMI ML 5 execution.

Our SCOE members are knowledgeable about the CMMI, International Organization for Standardization (ISO), and agile practices, and they are experienced with HVD. They develop improvement strategies for optimizing processes and value across the organization. Their combined expertise and experience help ensure more efficient operations and the use of industry best practices. They (1) provide support and consultation services to projects and management pertaining to the software development lifecycle and (2) guide projects to navigate new and changing customer requirements; in so doing, they help project teams deliver the highest quality work products.

The SCOE ensures that projects consistently execute at CMMI ML 5 and achieve results consistent with *ISHPI*'s track record for cost, schedule, and quality performance. It also helps ensure the project teams comply with contractual requirements for document standards and delivery schedules. Project teams also get help from the SCOE on project launches and project plans, quality plans, and CM plans. To inform this work, the SCOE and the project team members draw from the iPAL, which contains a 25-year accumulation of processes and best practices to enable rapid project estimation and start-up (Section 1.4).

The support provided by the SCOE for each project following the HVD methodology leads to disciplined software engineering because it

- develops workforce skills in disciplined software practices, including secure coding, estimation, design, and quality management
- ensures proactive creation of work product standards, guidelines, design patterns, and checklists, as well as consistent application of them
- systematically ensures application of secure coding practices and verifies that they have been followed
- continuously incorporates best practices from industry sources such as the Open Worldwide Applications Security Project (OWASP), the International Information System Security Certification Consortium (ISC),[2] and the Common Weakness Enumeration (CWE/SANS)

In addition, the SCOE brings optimization to the organization because it

- analyzes and gauges performance of the organization with respect to the organization's business objectives
- governs processes and work products, and leverages lessons learned across the organization
- initiates, tracks, screens, installs, and evaluates new methods and technology to improve the software engineering capability of the organization

- pilots and implements improvements without negatively impacting the organization
- maintains a consistent and persistent environment of continuous improvement across projects

1.3 Tailoring Practices to Meet the Specified Need

ISHPI understands that no two projects are alike, and that customer requirements and timelines differ and change from project to project. This is why we are flexible in our development approach. We adopt a realistic approach to each project based on time, requirements, and resources. Our implementation of our HVD process allows us to deliver quality products for both rapid development efforts and long-term projects alike. We have demonstrated this capability through our successful support of both commercial and U.S. federal government customers.

The Team Launch Process is the focal point for tailoring within the HVD process. With the guidance of a qualified team coach and *ISHPI* management, the teams follow the launch process to tailor based on customer goals, needs, constraints, and project attributes. The team produces the development strategy—considering alternatives including Spiral, Agile, and Waterfall—at the beginning of the project, and it takes ownership of the process that will be used in the project execution. The suitable process for the project is determined by systematically selecting and tailoring the applicable elements of the HVD toolbox to create a process and solution that is right for the individual project (Figure 4).

The process is reviewed by the project's SCOE QA representative to ensure that all mandatory tasks for quality and security are included in the process to achieve the best results for our customers.

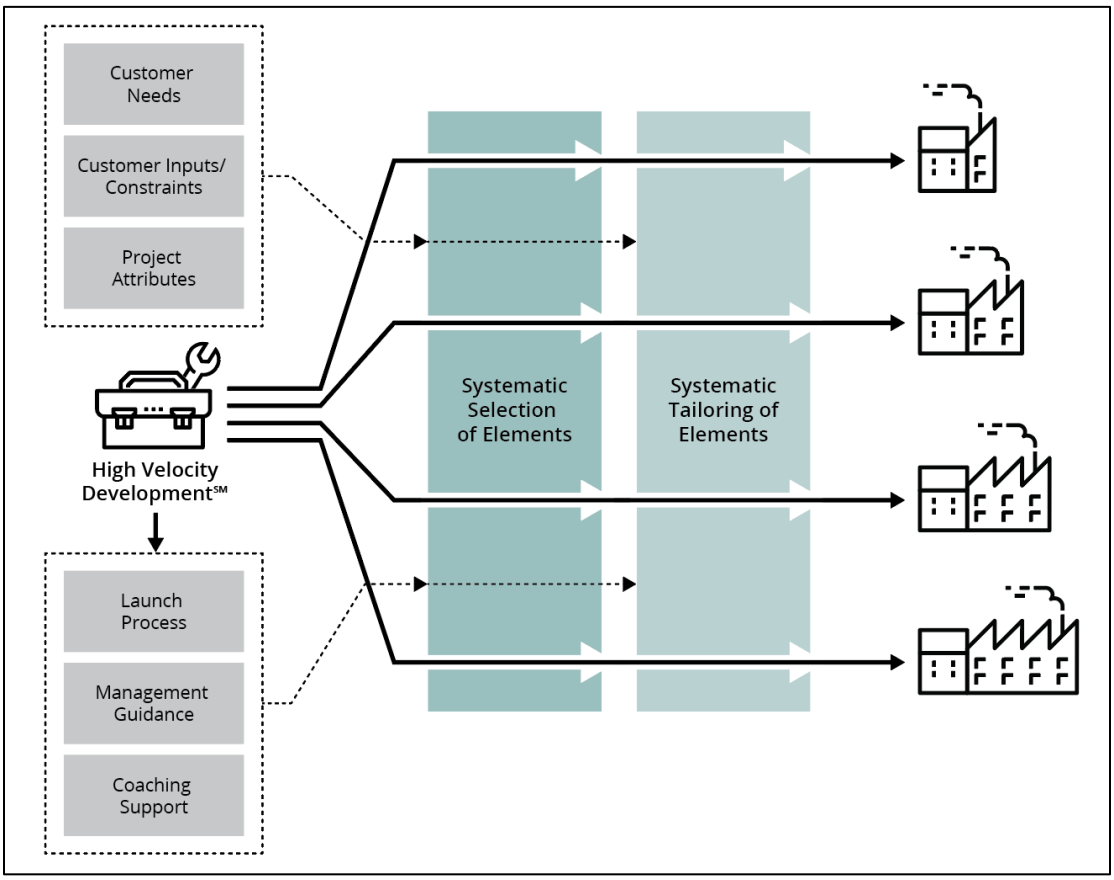

Figure 4: Tailor Practices to Meet the Specified Need

Our product delivery schedule has ranged from two to four weeks to twelve months, depending on the project and requirements, demonstrating our ability to tailor our processes to meet the demanding timelines and requirements of the customer. We have even tailored our process to accommodate a four-day rapid prototyping project with daily iterations, each encompassing planning, requirements elicitation, design, development, deployment, and customer demonstration.

Table 2 demonstrates the range of flexibility of our process, enabling us to effectively adapt to an assortment of customer and contract requirements.

Table 2: Examples of HVD Tailoring Flexibility on ISHPI Contracts

#	Contract	Key Tailoring Aspects
1	modernization contract	• performed estimating and planning at multiple levels of granularity, including milestone-level, annual contract level, and multi-phase budgeting • innovated processes to extract and validate requirements and business rules based on minimally documented code from a legacy system • accommodated multiple subteams concurrently and iteratively developing different functional areas of the system
2	maintenance contract	• conducted launches annually to address ongoing maintenance activities • performed kickoff meetings to plan each two-to-eight week enhancement with earned value tracking

#	Contract	Key Tailoring Aspects
		• defined a proactive monitoring regimen to ensure full availability of the system • customized the frequency of stand-up meetings, team status meetings, retrospective meetings, and causal analysis sessions
3	development, integration, maintenance, enhancement, and production support services contract	• adapted launches to accommodate multiple rounds of customer prioritization based on effort estimates and available budget • increased emphasis on peer reviews to stabilize code inherited from a previous contractor
4	maintenance and enhancement of third-party, custom-designed software contract	• adapted to needs for monthly releases • conducted launches quarterly, covering the next three to four releases • used the Redmine tool to track customer change requests • captured requirements details within individual change requests in Redmine since the inherited product had no documented requirements • adapted the design to focus on screen descriptions, data dictionary changes, and inline code comments for functional enhancements and refactoring

1.4 Process Improvement Proposals and *ISHPI* Process Asset Library

ISHPI follows our standard practice of leveraging data and feedback loops for continuous process improvement (Figure 5). As part of those practices, for each project we gather individual and team data and identify innovations, lessons learned, and process improvements. The mechanisms to capture improvements include data-driven retrospectives and causal analysis sessions that generate process improvement proposals (PIPs). Additionally, personal reviews, peer reviews, audits, and results of appraisals enable disciplined performance and provide continuous feedback on performance. These continuous improvement activities result in improved quality, improved productivity, and lowered cost.

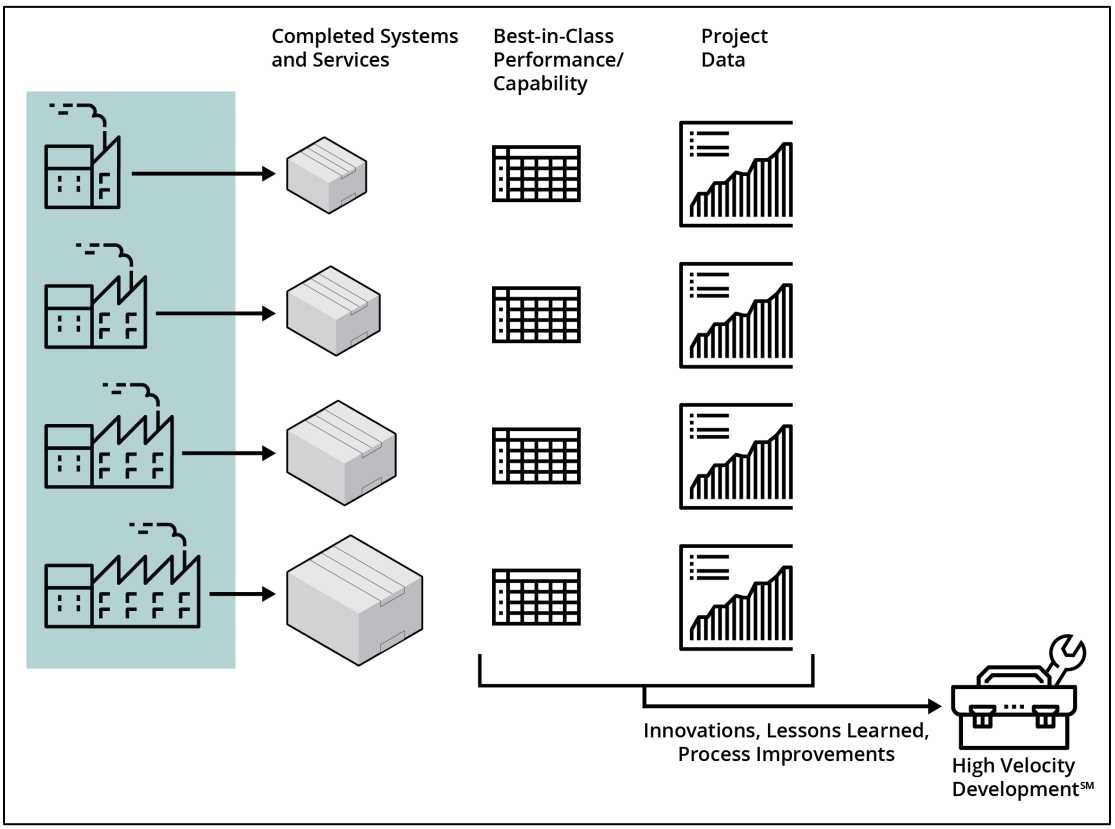

Figure 5: Exceptional Results with Feedback Mechanisms

HVD relies on processes and artifacts from the organization's readily accessible process asset library, iPAL. The repository includes over 400 process artifacts resulting from more than two decades of cumulative improvements using PIPs. The project team execution of component retrospectives, project causal analyses, and increment/project retrospectives are key triggers of PIPs. In addition, project teams capture best practices from tailoring (e.g., streamlining) and write PIPs to share these practices with the rest of the organization. Any staff member could also offer and document a specific suggestion for making the software process more effective. To date, the *ISHPI* AIS Division has written 2,012 PIPs, of which 1,351 were accepted, implemented, and deployed to the organization by the SEPG. Since these PIPs are identified, written, reviewed, and accepted by practitioners, the integration of these PIPs is almost seamless and easily institutionalized. These integrated improvements have evolved the *ISHPI* process from SW-CMM Level 1 to Level 5, and then to CMMI-DEV Maturity Level 5.

The SEPG receives, evaluates, implements, and disseminates improvements to process assets under its stewardship, driven by PIPs submitted by the project team members. The SEPG evaluates and implements PIPs that reduce cycle time, improve quality, improve predictability, reduce cost, and enhance user experience.

In our experience, when management provides an environment that recognizes and rewards quality work, developers improve continuously and deliver extraordinary results. We found it

effective to report the status on process improvement activities to senior management in Project and Process Status Review (PPSR) meetings and to make sure that those activities are funded.

iPAL provides an invaluable resource to all team members for secure coding practice resources, including standard lifecycles and work breakdown structures; secure coding standards; secure design patterns; templates; design checklists, which include items checking for secure design principles; and coding checklists, including items checking for well-known vulnerabilities.

ISHPI's iPAL enables rapid startup for our customers, and *ISHPI*'s emphasis on data, metrics, and continuous improvement activities results in higher quality, security, predictability, and productivity, as well as lower risk and cost for customers.

1.5 Key Differentiators of HVD

1.5.1 Agile Versus HVD Differentiators

HVD uniquely blends the responsiveness of Agile with the discipline of CMMI ML 5 practices. Unlike Waterfall, HVD is iterative, flexible, and proactively accommodates change. Some key elements of HVD include

1. Team launch

- We systematically plan team commitments by creating aggressive but realistic plans that are aligned and responsive to customer needs and goals.
- We make these commitments using historical data and involving customer stakeholders and all team members, which results in a shared vision and buy-in of the whole team.
- We systematically identify, evaluate, and plan for potential risks to generate appropriate mitigation actions.

2. Quality practices

- We consistently plan for and rigorously execute quality steps such as walkthroughs, personal reviews, peer reviews, and tests.
- We quantitatively plan for, monitor, and control quality, enabling us to focus on early defect removal and defect prevention.

3. Data-driven monitoring and control

- We track data at a granular level to provide a high degree of visibility into team activities.
- We use the data at the individual and team levels to gain a quantitative understanding of performance and status.
- We apply statistical models and projections to predict team performance relative to customer goals.
- We act proactively based on risks and the data to meet commitments and continuously improve capability and performance.

4. Disciplined engineering

- We train our team members in disciplined software practices, including estimation, design, quality management, and secure coding.
- We proactively create work product standards, guidelines, design patterns, and checklists and consistently apply them.
- We systematically apply secure coding practices and verify that they have been followed. These practices are informed by industry sources such as the OWASP Top 10 project, the (ISC)^2 Common Body of Knowledge, and CWE/SANS Top 25 Most Dangerous Software Errors.

5. Collaborative and iterative execution

- We work closely with our customers to deeply understand their business and needs. We use that knowledge to deliver superior solutions that are right the first time. We welcome and promptly incorporate changes and maintain awareness of evolving customer needs and priorities so that we can rapidly respond.
- Our team members freely collaborate with each other, and our self-managed teams take corrective action without the need for management intervention.
- We develop iteratively and produce frequent testable milestone deliveries to foster a shared vision, maximize value, and reduce risk.
- We proactively share information about team activities and status with our customers to avoid surprises.

6. Continuous improvement

- We coach our teams by providing experienced experts that perform analyses, provide insights, and offer suggestions that maximize individual and team performance.
- We routinely analyze data and leverage our experiences to systematically identify and apply lessons learned at individual and team levels to improve practices that benefit future components, iterations, and projects.

Table 3 provides a comparison of Agile/Scrum and Agile HVD attributes based on the key elements of HVD.

Table 3: Comparison of Agile/Scrum and the Agile HVD Software Development Approaches

Attribute	Agile/Scrum	Agile High Velocity Development℠
Satisfying the customer	Early and continuous delivery	Incremental high-quality deliveries of functionality driven by customer needs with minimal technical debt Benefit: software that meets customer needs, delivered incrementally and right the first time
Responding to change	Welcome change	Change requests welcomed and seamlessly incorporated using flexible, disciplined change management practices Benefit: thorough impact analysis and prompt incorporation of changes to deliver working software right the first time and meet customer needs

Attribute	Agile/Scrum	Agile High Velocity Development℠
Frequency of delivery	Deliver frequently, preferring shorter timescales	• Iterative development and delivery with flexible durations • Duration of each iteration based on customer needs and team strategy Benefit: flexibility to formulate the most efficient development approach to meet customer needs
Working with the business	Work together daily	• Frequent customer collaboration and stakeholder involvement • Collaboration practices that efficiently focus on obtaining a deep understanding of the customer's business Benefit: efficient use of customer/stakeholder time while working with the business to build software that meets customer needs
Empowering individuals	Support motivated individuals	• Individuals trained in disciplined software practices including estimation, design, quality management, and secure coding • Repeatable practices for quickly building and supporting self-managed teams motivated by shared goals and defined roles • Coaching for individual team members and the overall team Benefit: cohesive teams operating at peak performance, working efficiently to successfully achieve customer goals at the lowest cost
Communications	Face-to-face conversations	• Proactive and frequent collaboration with customers and team members • Technology supplements face-to-face communication when needed • Both informal and structured/facilitated communication, when appropriate, that focuses on obtaining a deep understanding of the customer's business Benefit: efficient, effective use of everyone's time—both customer stakeholders and the team
Measuring progress	Working software	Leading indicators that provide accurate, data-driven visibility into the team's progress Benefit: on-time delivery of working software without accumulating technical debt
Project pace	Maintain a constant pace	• Pace driven by the team's quantitative understanding of its capabilities based on historical data • Detailed plans for individual iterations, with high-level commitments for increment releases during project launch Benefit: teams consistently meeting aggressive but realistic commitments

Attribute	Agile/Scrum	Agile High Velocity Development℠
Technical excellence	Focus on technical excellence and good design	• Focusing on architecture early in the project to build a foundation for solid design • Consistent application of work product standards, guidelines, design patterns, and checklists • Systematic application of secure software principles throughout the lifecycle Benefit: well-designed, highly maintainable secure software with minimal "churning," leading to the lowest total cost—both during development, and during operations and maintenance
Simplicity	Maximize the amount of work not done	Relentless optimization by focusing only on tasks and work products that add value Benefit: lowest cost
Team organization	Self-organizing teams	• Repeatable practices for quickly building self-managed teams motivated by shared goals and defined roles • Team collaboration using processes that guide the team to consider every aspect of the problem and design the best solution Benefit: superior technical solutions
Improvement	Periodic reflection and adjustment	• Focusing on continuous improvement at the component, increment, project, and organization level • Data-driven causal analyses and retrospectives to generate proposals that systematically improve practices for future iterations and projects Benefit: lowest risk to scope, budget, schedule, and quality by using proven practices that have been finely honed over decades
Additional Agile HVD Attributes		
Data-driven	• Granular data (size, effort, schedule, cost, quality) to provide high visibility • Quantitative understanding of individual and team performance • Statistical models and projections Benefit: teams consistently meeting commitments	
Built-in quality	• Early defect removal and prevention achieved through walkthroughs, personal reviews, and peer reviews • Comprehensive testing • Quantitative planning and control of quality Benefit: secure, working software, right the first time with the lowest cost	

1.5.2 Self-Managed Teams

The HVD process entails the following practices and behaviors to leverage the power of self-managed teams to meet objectives:

- All team members participate in the Team Launch process and take ownership of the project plan, process, and commitments.
- Team members are trained to make plans using personal historical data to estimate effort and the schedule.
- Team members have the conviction to defend their plans and negotiate an aggressive and realistic schedule.
- Teams practice early defect removal using personal reviews and team peer reviews.
- Teams put the highest quality product into test and eliminate the long test and rework cycle that causes schedule slippage in software projects.
- Teams track progress weekly using earned value management with the ability to detect as little as a one-day slip in schedule and take corrective action.
- Teams identify the top risks that could impact cost and schedule adversely and track mitigation actions weekly.
- Teams have mentors who motivate and coach individuals to achieve their own optimum performance.

1.5.3 Continuous Improvement

Continuous improvement is integrated into project execution by the following HVD practices:

- We capture and analyze data after each development cycle.
- We build personal history from estimated and actual data for size and effort.
- We apply regression to improve estimation.
- We analyze defects and implement action items to prevent similar defects in the future.
- We coach our teams by providing experienced experts who perform analyses, provide insights, and offer suggestions that maximize individual and team performance.
- We routinely analyze data and leverage our experiences to systematically identify and apply lessons learned at individual and team levels to improve practices that benefit future components, iterations, and projects.

For example, Figure 6 shows effort estimation error for each component tracked for a team member before and after performing causal analysis on effort estimation error. The following action items were identified and implemented as part of the causal analysis meeting:

- Attend overviews to understand the functionality of the unknown codebase and/or technology.
- Simplify the personal review checklist.
- Review critical design standards and samples.
- Discuss test scenarios before writing test cases.

As a result of implementing these action items, the average estimation error decreased from 22.6% to 5.96% (Figure 6). However, team members noticed that the average estimation error is increasing again (from component 263 onwards), which prompted them to conduct another causal analysis session to identify root causes for the change and action items to implement to reduce the average estimation error and variation.

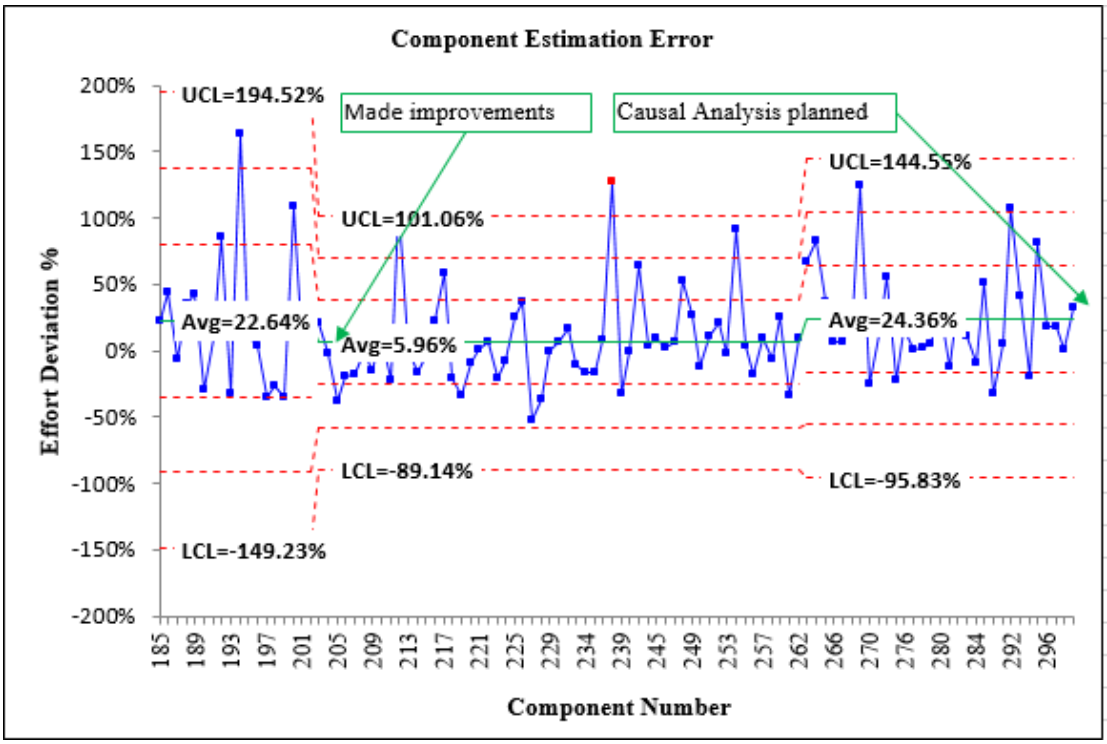

Figure 6: Component Estimation Error

1.5.4 Mentoring and Coaching

Team coaching is a critical element of HVD. The team coach provides ongoing mentoring for the project team members to maximize the benefits of the available individual, team, and organizational data. Our coaching approach focuses on continuously improving the predictability, quality, and efficiency of each individual and the team as a whole. The team coach role within the HVD method enables individuals and teams to execute at their highest possible level of performance.

This technique systematically builds jelled, disciplined, self-managed teams composed of individuals dedicated to personal excellence who take ownership and responsibility for the quality of their work products. For over 20 years, *ISHPI* AIS has successfully applied this coaching model to

- support team members with analysis of their performance data
- assist individuals and teams in setting specific, measurable goals for improvement
- mentor team members in the effective use of the HVD method
- provide expertise in industry best practices and advanced data analysis techniques

- provide independent perspective, suggesting improvements and optimizations
- facilitate project launches, causal analyses, and retrospectives that guide team members in completing the meetings with maximum focus, efficiency, and productivity

2 Application of Quantitative Techniques in HVD

The *ISHPI* AIS Division's strategy is based on AIS's balanced scorecard. The scorecard identifies objectives, core outcomes, and performance drivers for financial, customer, employee, internal business process and learning and growth perspectives.

Controlling the *ISHPI* AIS Division's process performance based on the balanced scorecard objectives involves taking measurements, analyzing these measurements, and making adjustments to maintain process performance within acceptable limits. When the process performance is stabilized, the organization's defined software process, the associated measurements, and the acceptable limits for the measurements are established as a benchmark and used to control and improve process performance quantitatively.

The SEPG is the primary entity responsible for coordinating the organizational process performance and quantitative project management activities for the organization.

2.1 Principles Driving HVD Metrics Framework

In 1992, *ISHPI* AIS established a business goal to "Improve profitability and customer satisfaction by delivering substantially defect-free products on predictable cost and schedule" [Perini 2016]. At the organization level, we started with the goals and then used the Goal-Question-Metric (GQM) paradigm to derive the metrics that would help us achieve the goals [Basili 1994]. We describe the measures that our software application teams collect, analyze, and report to fulfill these goals. These measures are proven to help them develop and deliver software with little or no defects or vulnerabilities.

The *ISHPI* AIS Division principles that drive our metrics framework are the following:

1. **Quality and reducing vulnerabilities should be the number one goal for every software team.**

   It is well known that some software defects result in vulnerabilities. To reduce vulnerabilities, software teams need to be focused on defect injection and removal throughout the software lifecycle. Teams need to be aware of where the defects are injected and where they are removed. Teams cannot rely on testing alone to find and remove software defects, including security vulnerabilities. Instead, teams need to focus on identifying and removing defects as early in the lifecycle as possible. Our experience has shown that many CWEs, such as buffer overflows, cross-site scripting, and failures to validate input values, can be easily found and corrected in personal reviews and peer reviews of the source code. Improving quality by not relying on testing alone addresses many of the software security issues [Seshagiri 2014].

2. **Software teams should put the highest quality code into test.**

   Large systems are built by integrating small components developed by individual developers. If the components are of poor quality, the delivered product will have a large number of

defects and vulnerabilities. Teams need to make sure individual developers take pride and responsibility for the quality of their work products. The developers should be trained in and supported by an agile, disciplined, personal development process that should enable collection, analysis, and reporting of defect injection and removal data through code completion. The developers need to make sure that the components are defect free at completion of component development. This will make more time available to deal with defects and vulnerabilities in technology and system-level stacks, which in turn significantly reduces total development time. Teams should include defect data, along with cost and schedule data, in weekly status reports to development and customer management. The data collected needs to be both precise and accurate. Team members need to be trained in data collection and analysis so that they can effectively use these metrics to be self-managed and take timely action to meet personal and team goals.

3. **Unless data is collected and used by trained team members, it won't be useful.**

   Software teams are often working to meet arbitrary and unrealistic delivery commitments imposed on them. Team members need to be trained in estimating sizes of software components and negotiate aggressive and realistic delivery commitments based on personal and team historical data that shows the relationship to size and effort. Data that clearly shows the impact of disciplined practices, such as personal reviews and peer reviews on reducing rework and total development time, helps the team and individual developers to make responsible delivery commitments.

   Software teams are expected to make frequent deliveries of working software. Imposing time-consuming data collection of questionable value is counterproductive. *ISHPI* AIS software teams collect only size, task time, defect, and task completion data, as shown in Table 4.

   Table 4: Base Team Measures

Measure	Unit
Size	Lines of code
Task time	Minutes
Defects	Number
Task completion	Yes/no

   For the resulting data to be precise and accurate, teams need to have a clear definition of the measures. For example

   - Size measurements need to be based on defined coding standards and line-of-code counting rules.
   - Tasks need to be granular, and measurements for each task should track only the time spent on it, excluding interruptions.
   - Defects need to have standardized defect types based on orthogonal defect classifications.
   - Task completion needs to be based on an agreed "definition of done" at the task level.

From the precise and accurate base measures, the "vital few" performance metrics are derived to help self-managed teams consistently deliver substantially defect-free software on predictable cost and schedule. In turn, management is assured that the organization's business model of firm fixed price with guaranteed quality results in customer satisfaction and company profitability. This leads to the final principle.

4. **Software teams must be focused on the "vital few" metrics and not the "trivial many."**

   Our experience has shown that self-directed teams consistently collect, analyze, and report the "vital few" metrics and deliver software that is substantially free of defects and vulnerabilities on predictable cost and schedule.

   Management bears the responsibility to do the following:

   - Staff projects with team members who are trained and/or coached in estimating, planning, tracking, measuring, and managing quality.
   - Start each project right with a cohesive team that maintains a shared vision and makes disciplined commitments with the assistance of a coach.
   - Support the teams in collecting, analyzing, and reporting product and process data (size, effort, schedule, and quality).
   - Provide the teams data needed to make decisions and improve continuously through causal analysis and retrospectives.
   - Ensure team members are trained to conduct peer reviews of design and code artifacts to put the highest quality code components into test, striving for 90% to 100% defect-free components with zero cybersecurity vulnerabilities.
   - Ensure the review checklists incorporate specific checks for the types of vulnerabilities identified within the OWASP Top Ten and Common Weakness Enumeration (CWE/SANS) Top 25 Most Dangerous Programming Errors.
   - Ensure code vulnerability analysis is performed.
   - Require teams to report status weekly, with precision and accuracy, monitoring actual numbers of defects throughout the software development lifecycle and enabling the team to proactively manage the quality.

5. **Vital few performance metrics are tracked.**

   We discuss the vital few performance metrics that really matter and help software teams manage the software work by managing quality. Table 5 shows data collected for the vital performance metrics at the component, increment, and project level. The data from projects across the organization are used to derive *ISHPI* AIS organizational performance metrics, as shown in Table 8. Table 6 contains the operational definitions for these metrics.

   These metrics help individuals at the component level, the team at the increment and project level, and the SEPG at the organization level to guide the course of action towards the achievement of the goals or to help to evaluate the result of the actions.

   The metrics identified as leading indicators in Table 5 are proactively monitored to provide an early indication of whether the strategy is being implemented successfully to build a

secure, high-quality product. Corrective actions are taken as needed based on these leading indicators. These corrective actions have a significant impact on the lagging indicator metrics. The lagging indicators measure the results of the practices used by the individual, team, and organization. Lessons learned from both positive and negative outcomes are analyzed and used for continuous improvement of the individual, team, and the organization [Seshagiri 2015].

Table 5: Vital Few Performance Metrics Tracked

Performance Metrics	Leading Indicator?	Lagging Indicator?	Organization	Project	Increment	Component
Planned vs. actual size (for primarily new development)	Y			✓	✓	✓
Planned vs. actual effort	Y		✓	✓	✓	✓
Planned vs. actual schedule with the planned scope delivered	Y		✓	✓	✓	✓
Planned vs. actual earned value	Y				✓	
Total Cost of Quality (COQ)	Y	Y	✓	✓	✓	
First Time Right (in acceptance test): number of changes with no acceptance test defects		Y	✓	✓	✓	
Acceptance test defect density in delivered code (for primarily new development)		Y	✓	✓	✓	

Table 6: Operational Definitions

Term	Definition
% Schedule Deviation	Actual schedule divided by planned schedule (in days) times 100 minus 100
Task Time	A defect is found during the acceptance test of the release/application. This defect may be identified by the customer as well as the *ISHPI* team. The acceptance test period starts after the completion of the system test of the release/application and ends when the code is deployed to production.
COQ	COQ= (effort in appraisal tasks + effort in prevention tasks + effort in failure tasks) x 100 Total effort towards the commitment (including project management) Appraisal tasks: personal reviews, peer reviews, and first-time unit, integration, and system test execution Prevention tasks: training, retrospectives, and causal analysis Failure: analyzing and fixing defects found in reviews and testing (rework effort)

Term	Definition
Defect Density	The number of defects identified in a product divided by the size of the product, expressed in standard measurement terms for that product (e.g., 1000 lines of new and changed code)
Earned Value	Percentage of effort for tasks completed divided by total effort of all tasks (excluding ongoing tasks)
First Time Right (in acceptance test)	Deployed software changes, accepted by the customer the first time, with no further rework

2.2 Decision Process Based on Metrics and Measures

ISHPI AIS teams use metrics and measures in a multitude of ways to manage their work and meet project goals and customer objectives. Secure software that is free from vulnerabilities starts with responsible team commitments—whether for an individual software component, an increment, or a longer term timeline. When teams are under pressure to meet what turns out to be an impossible commitment, they often fall victim to taking shortcuts, which degrade quality, create technical debt and rework, and open the door to security vulnerabilities. When team members do not understand the relationship between time and size, it is hard to make commitments they can fulfill. Therefore, individuals and teams need to keep a historical record of their estimates and actuals, enabling them to make aggressive but realistic commitments for future efforts.

Teams do not review the data in isolation—they always use it in combination, and with a focus on the bigger picture. Data falling outside the expected range is not necessarily a cause for alarm: Teams use such data as a trigger to dig deeper, perform more analysis to understand the circumstances, and make decisions that mitigate risks and address issues. Here are examples of some of the measures and metrics we review to make decisions while managing our projects:

1. **Planned versus actual size, planned versus actual effort, and planned versus actual schedule**: During component retrospectives, team status meetings, and increment retrospectives, the team reviews the size, effort, and schedule deviation data. Based on the data, the team makes a judgement about the need for corrective action. If deviations are significant, then the team assesses the causes and potential impact on schedule commitments for the committed scope. If corrective action is required, some possible actions include renegotiating the schedule, deferring functionality, and adding staff. The team identifies lessons learned based on the causes of the deviation and submits improvement suggestions to be applied to future increments and projects.

2. **Planned vs. actual earned value**: During team status meetings, the team reviews earned value deviation data and makes a judgement about the need for corrective action. If deviations are significant, then the team assesses the causes (e.g., quality issues) and potential impact on schedule commitments for the committed scope using earned value projections and “what if” scenarios enabled by our process performance model. If corrective action is required, some possible actions include strengthening peer review practices, renegotiating the schedule, deferring functionality, and adding staff.

3. **Total Cost of Quality (COQ)**: During team status meetings and retrospectives, the team reviews the planned, actual, and projected COQ tables and charts: appraisal plus prevention, failure, and total. After reviewing the data, the team determines whether the project is on course to be completed with a COQ that is too high and, thus, requires corrective actions. If so, some possible corrective actions include
    - proactively monitoring and analyzing the factors that affect the appraisal plus prevention and failure COQ; these factors include
        - for appraisal COQ: slow review rates, slow test execution, and poor product quality (Poor quality product slows the review process.)
        - for prevention COQ: training to address new technologies, training new team members, and time spent in causal analyses and retrospectives
        - for failure COQ: rework for the defects found in review and test activities
    - conducting causal analysis of selected failure outcomes and identifying action items
    - re-evaluating after corrective actions have been put into place
4. **First time right (in acceptance test)**: During increment retrospectives, the team reviews the number of features or changes that were incorporated into the product "first time right." The team analyzes whether the outcome was within the expected range, and the team identifies lessons learned and submits improvement suggestions to be applied to future increments and projects for both positive and negative outcomes.
5. **Acceptance test defect density in delivered code**: During increment retrospectives, the team reviews acceptance test defect density. The team identifies lessons learned based on the causes of quality issues (as well as quality successes) and submits improvement suggestions to be applied to future increments and projects. The team performs formal root cause analysis on any individual defects (or related groups of defects) that escaped to the customer to identify specific improvements that would have prevented the defect from escaping.

2.3 Data Collection Framework

HVD uses CMMI, TSP, PSP, and other agile practices to collect data at the individual, project, and organizational levels. The data is collected by the individuals at the project level and submitted to the SEPG to include it in the organization repository. The data from the organization repository is used by the teams, as applicable, during the estimation and planning stages of the software development lifecycle. Data collection and lessons learned take place during the execution and the retrospective stages of the software development lifecycle. Data quality is reviewed and monitored periodically—as frequently as every week at the project level. The SEPG reviews data quality when the teams submit project data for each release and/or increment plan and at the end of the contract year. All metrics and analysis are dependent on the four fundamental measures of size (e.g., lines of code), effort (hours), schedule (task closure), and defects as described in Section 2.1.

2.4 Statistical Techniques

ISHPI's AIS Division has incorporated a variety of statistical techniques into HVD to provide value to the organization and our customers. We apply those techniques at various levels: Most importantly, individual team members (e.g., software engineers) and project teams use them in their day-to-day work to accomplish their project goals. In addition, our SEPG uses statistical techniques at the organizational level to perform analysis across projects and understand our overall capability. Although the SEPG provides guidance and support, the gathering, analysis, and use of measurements and metrics is not constrained to the SEPG: These activities permeate our culture at all levels of the *ISHPI* AIS Division [Shook 2023].

The foundation of HVD is fundamentally quantitative, and use of the statistical techniques within HVD provides *ISHPI* with a deep understanding of our capabilities. Our practice is to gather data consistently, use that data, learn, and continuously improve. However, in so doing so we maintain a focus on business value:

- We employ our capability to make responsible commitments for effort, schedule, and cost using our estimation process.
- We employ our capability to manage quality using our quality practices—especially peer reviews.
- All of this enables *ISHPI* to provide a positive experience to our customers by consistently meeting our commitments and consistently delivering our work products first time right.

2.4.1 Object/Component Size Database

The *ISHPI* AIS Division creates and maintains object/component size databases to understand the actual sizes of work (e.g., lines of code) performed in the past to estimate the effort more accurately for future planned components. Most commonly, each project team instantiates its own object/component database at the beginning of the project. They accumulate the actual sizes as work is completed. The team refers to the resulting historical list of sizes of work it has performed when it plans each subsequent project increment. In addition, our SEPG maintains an organization-level size database, which is available for all teams to reference. At project milestones (such as at the end of each project increment, project phase, contract period of performance, or project/contract), each project team performs an SEPG Data Collection process to submit the team's actual size data to the SEPG, and the SEPG adds it to the organization size database. Newly formed teams refer to this organization size database to help them plan work before they have accumulated any of their own project-specific data.

Within HVD, a **component** is a logical chunk of work that is the responsibility of a single author (although other team members are typically involved to perform walkthroughs and peer reviews and provide other support). Components typically follow a relatively consistent lifecycle pattern that, for coding components, typically includes planning/analysis, design, development, peer review, unit testing, and a component-level retrospective. Teams tailor the component lifecycle when needed, based on the nature of work to be performed, but otherwise try to remain as consistent as possible. For a project increment, the team decomposes the required work into components at a "reasonable" level of granularity—typically the amount of work the author can

complete in a week or two. However, the team must identify components in a way that reflects the way the work will actually be performed and provides a clear "definition of done." Ultimately, the team must break its work down into components in a way that is useful to the team, and this usually results in useful historical data.

Size Measures: Sizes of items placed in the object/component size databases are usually measured as either new and changed lines of code (LOC) or simply effort (hours) but can also be other measures, such as pages for non-coding types of work products. Team members measure the components' actual LOC using a tool (typically Beyond Compare) during component retrospectives. *ISHPI* teams follow documented guidance for consistent LOC counting, tailored from organizational guidelines provided by the SEPG, to ensure consistency across project team members and across projects. This guidance includes tool settings for applicable languages and typically excludes automatically generated code.

We prefer to use a countable size of physical output produced during the component, such as LOC, whenever possible, and we use LOC when component sizes are proportional to the effort needed to perform the work (i.e., accurately estimating the size of a new component helps to accurately estimate the effort). However, in our experience, for some types of work, size and effort are not proportional (i.e., knowing the size of what we are going to produce does not help us to accurately estimate the effort needed to complete that work). We have found that this is especially the case with software maintenance work, such as implementing a small enhancement or fixing a reported defect. In these cases, the size of the new and changed LOC is often extremely small, but the effort to determine the exact code to change in a vast code base is often significant (and not proportional to that size). Similarly, we have found that actual effort is also not proportional to the size of the *base code* that is analyzed for potential impact. Since such maintenance work has tremendous variability, and effort is neither proportional to the base LOC nor new and changed LOC, we have found that size is not useful for estimation purposes in this context. In these cases, we simply use effort when physical size does not make sense, and the sizes in the object/component size database are simply the effort (hours)—the actual effort to complete the defined lifecycle of the component.

When estimating a new project increment, our project team members identify objects, components, or activities that need to be completed to accomplish the objective. These are often in the form of one or more user stories that usually identify the development of new functionality, an enhancement, or a change to fix a latent defect. For each component, the team references the database to identify previously completed components of similar magnitude and complexity and notes their *relative size*s, sometimes called "tee shirt sizes"—very small (VS), small (S), medium (M), large (L), or very large (VL). These relative historical component sizes are determined by the distribution of the actual measured sizes, as discussed below, and each relative size has size value. Using that historical data, the team uses its judgement to choose the best relative size for the new component being estimated. The team estimates the size of the new component using the size corresponding to the selected relative size.

ISHPI's HVD process provides guidance that estimates should be based on previous empirical historical data, whenever possible. The resulting practice resembles the "planning poker" approach used by teams following other Agile methodologies. However, instead of selecting "tee

shirt sizes" that correspond to a pattern, such as a Fibonacci sequence, our teams use sizes that reflect the actual size distribution of work previously performed by our teams and team members. This technique works well for our teams, enabling them to consistently produce very accurate effort estimates and make responsible scope and schedule commitments, especially when combined with other statistical techniques embedded within HVD.

Calculation of Relative Sizes: Based upon empirical historical data, *ISHPI* generally establishes the sizes for each relative size category as follows:

- Medium: mean size of the actual historical component sizes
- Small and Large: mean +/- 1 standard deviation of the actual historical component sizes
- VS and VL: mean +/- 2 standard deviations of the actual historical component sizes

However, since measured component sizes are often somewhat small but, importantly, can never be negative, we do not expect the measured size data to follow the normal distribution. Instead, we expect them to be approximately log-normally distributed [Humphrey 1995b]. Therefore, we apply a simple mathematical transformation to the data within our size databases to account for this. When calculating the relative size values, we apply a simple algorithm (Figure 7):

- Calculate the natural logarithm of the size of each historical component (so that resulting values much more closely resemble the normal distribution).
- Calculate the mean and standard deviations of those adjusted values.
- Calculate the relative size values as the mean +/-1 and +/- 2 standard deviations (σ).
- Apply the inverse "natural exponential" function to each relative size to return to the scale of the original data.

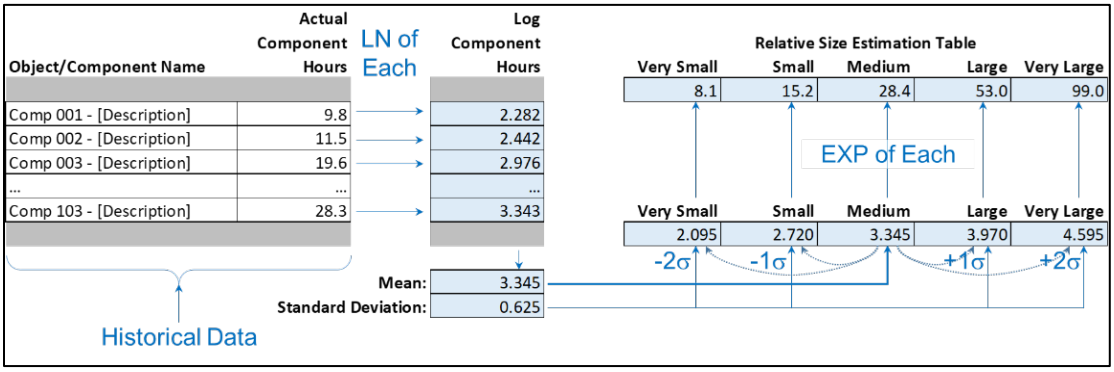

Figure 7: Object/Component Size Database and Relative Size Calculation

Over the past 25 years, our teams have found this method to be straightforward and effective, and the technique works equally well with physical size measures (e.g., LOC) as with effort. *ISHPI* teams have experienced that the historical size data is indeed similar to a log-normal distribution. For example, from the actual historical data of one of our teams, we see the raw (untransformed) distribution (Figure 8) and the transformed distribution after applying the natural logarithm to each data point (Figure 9).

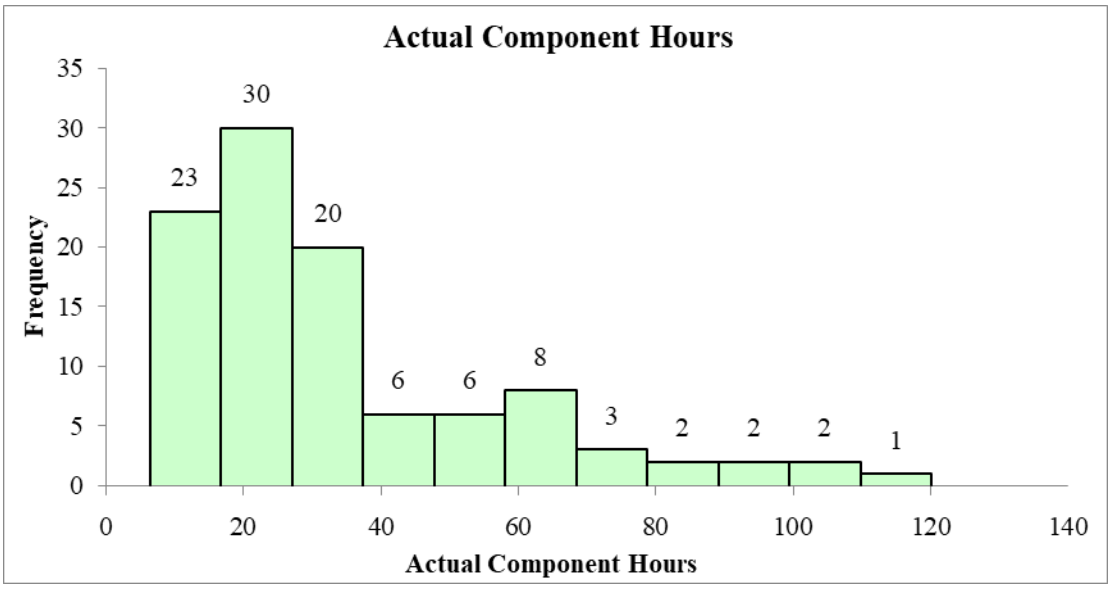

Figure 8: Raw Actual Component Hours in Object/Component Size Database

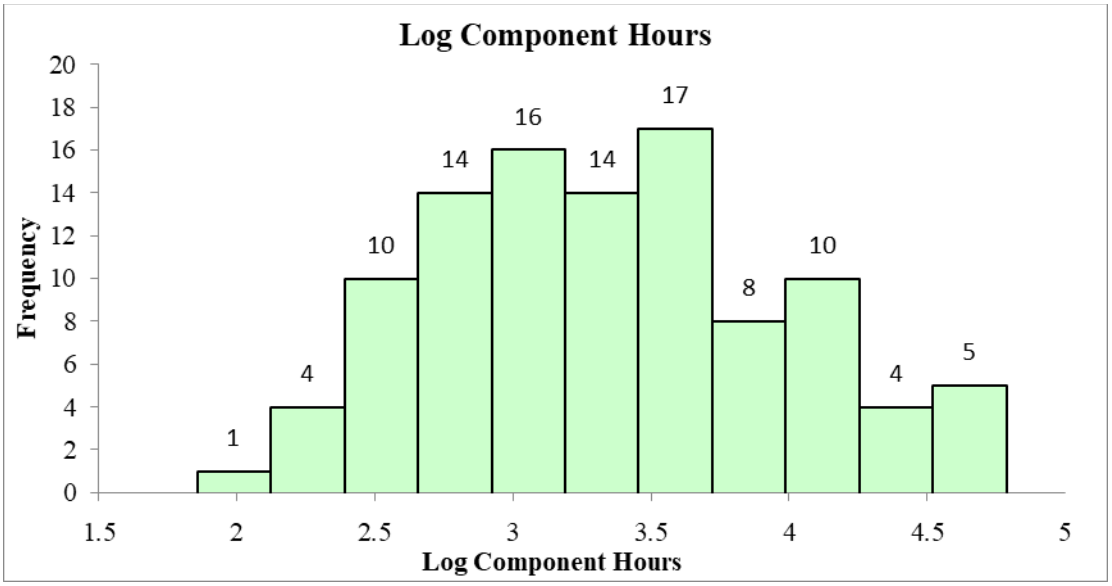

Figure 9: Log Transformation of Component Hours in Object/Component Size Database

Within our object/component size databases, *ISHPI* separates our historical data into different subgroups for different types of work, since they have different size distributions. For example, we routinely consider the following attributes to determine the placement of component data into subgroups:

- Size measure: Separate components measured only in hours versus those measured in LOC, for instance.
- Programming language: Separate Java components from C#.NET components, for instance.

- Architecture level: Separate user interface, middle tier, and back-end database components into different subgroups.
- Lifecycle: Separate components that have substantially different component lifecycles into different subgroups.
- Development type: Separate new development and maintenance into different subgroups.

Our object/component size database tools conveniently support this stratification into subgroups. Although subgrouping is particularly important at the organizational level (for historical data collected by the SEPG and provided to new project teams), individual project teams rarely need to worry about more than a few subgroups for their own data.

2.4.2 Linear Regression

ISHPI uses linear regression to understand the correlation between two data sets. The "best fit" regression line formula is $y = \beta 1\ x + \beta 0$. *ISHPI* uses linear regression during general data analysis in many ways at the individual level, team level, and organization level. However, the most important application within HVD is to correct for estimation bias at the individual/component level, team/activity level, and project level. For this application, we maintain historical data of estimated versus actual values for each item. For example, Figure 10 shows the relationship between estimated effort and actual effort for over 100 components developed by a project team. When we estimate, we apply the $\beta 1$ multiplier and $\beta 0$ constant to the new raw estimate ("x") to derive an adjusted estimate ("y"), which should be more accurate and correct for any recurring tendency to overestimate or underestimate. When we apply this method, we make sure to choose historical data that is similar to the future work being estimated.

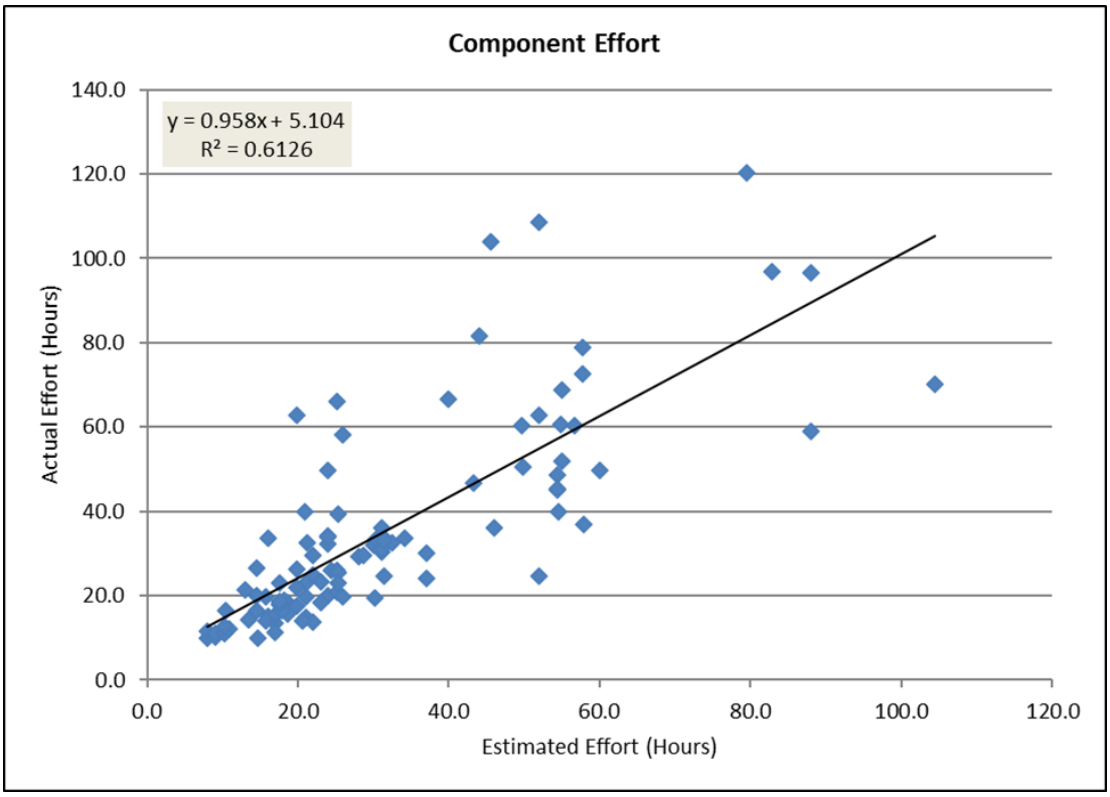

Figure 10: Estimation of Component Effort

When we apply linear regression to correct estimates, we consider several factors to ensure it makes sense to do so. Specifically, we always calculate the correlation coefficient (R^2 value) and consider the significance and linear regression coefficients (β1 and β0). For example

- **R^2 value:** An R^2 value of 1.0 indicates a "perfect" linear correlation. We consider values greater than 0.9 to be a very strong correlation, at least 0.7 preferred for use in planning/estimation, and at least 0.5 to be used with caution. If the R^2 value approaches 0.5 or lower, we do not use regression and instead revert to some other estimation correction method, like "averaging."
- **Significance:** One data pattern we commonly see is when the estimated sizes of components are quite tightly clustered with a relatively narrow range of values (i.e., the estimated maximum value is close to the estimated minimum value). The natural variation in actual sizes for those components tends to result in a "shotgun blast" pattern. Here, one could plausibly draw a line through the data points in just about any direction: There is no particular "best fit" line. In this scenario, the R^2 value is typically much too low to be acceptable. However, if there are a small number of data points outside the cluster, those data points tend to have disproportionate "leverage" that establishes a very clear regression line (and therefore a relatively high R^2 value). Regression in such situations needs to be used with caution, and this shows up with a low statistical significance value—especially with a small number of historical data points. Again, in such situations, we would not use regression and instead revert to some other method.

- **β1 coefficient:** We consider whether the β1 coefficient "makes sense." Generally, we expect β1 to show a modest positive slope. For example, when correcting an effort estimation, we would generally expect β1 to be 0.9 <= β1 <= 1.1 (effectively adjusting the "x" value by up to +/- 10%). When the β1 slope is too steep, too flat, or negative, we typically do not apply regression. (That is, we avoid scenarios that do not make sense like, "it doesn't matter how big you think it is, it will take the same amount of effort" or "the bigger you think it is, the less effort it will take.")
- **β0 coefficient:** We do not typically apply regression when the β0 constant is quite large compared to the estimate itself (i.e., the "x" value), which would be interpreted as huge amounts of fixed overhead, which may not make sense.

2.4.3 Prediction Intervals

When *ISHPI* teams produce effort estimates, create schedule plans, and make milestone commitments using linear regression, we have the opportunity to extend the linear regression technique using prediction intervals. With this technique, we not only adjust the raw estimate to correct for estimation bias, but we also analyze the *range* of expected results around the adjusted estimate based on our relevant historical data. The use of linear regression together with prediction intervals comprises a process performance model (Section 2.4.4).

The prediction interval analysis enables *ISHPI* managers to make an informed decision about how aggressive our estimate is and how much risk we are taking on (or want to take on). Use of this technique contributes to our ability to make responsible commitments and consistently meet (or beat) our commitments for effort, schedule, and cost.

We calculate prediction intervals based on the statistical distribution of our historical data. This calculation results in the upper prediction interval (UPI) and lower prediction interval (LPI) values for the estimate, between which the actual result would likely fall with the targeted confidence (Figure 11). When we apply the prediction interval, we most often choose a 70% confidence interval (which implies that 85% of the data points would likely fall below the UPI), but we can choose any prediction interval we want, based on the nature of the commitment we are making. Because prediction intervals are built upon and extend linear regression, all the caveats and constraints of linear regression apply (as described in Section 2.4.2).

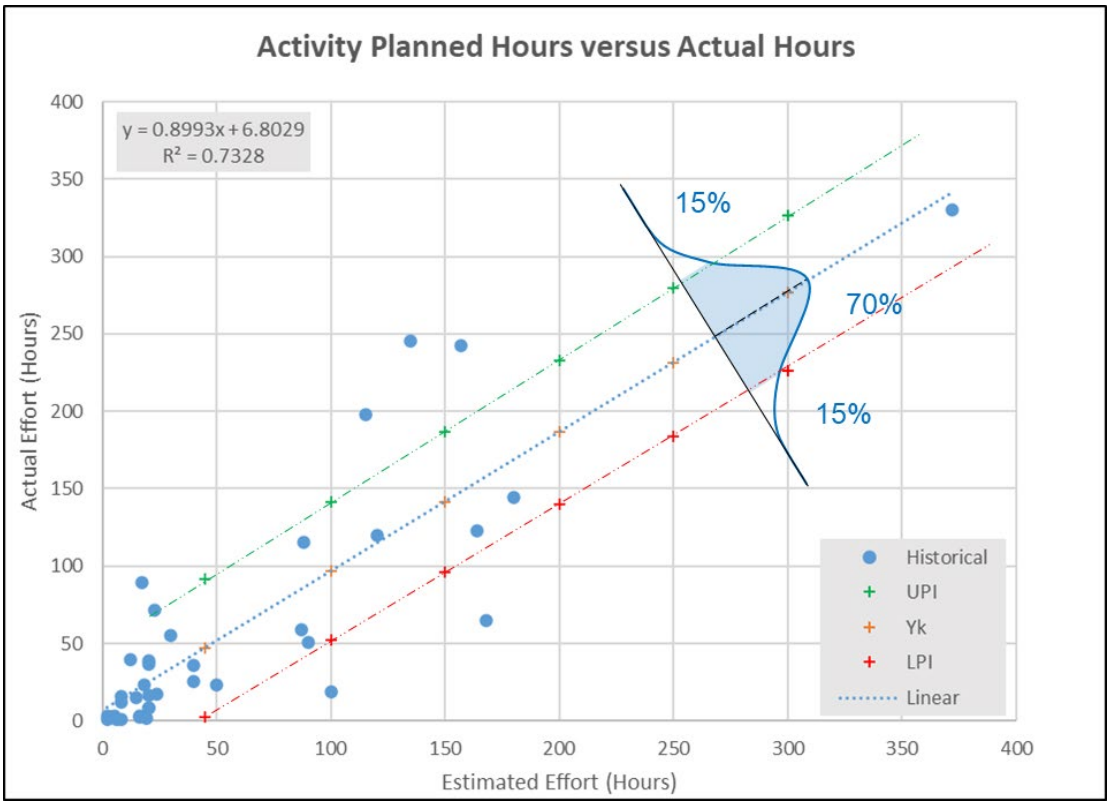

Figure 11: Prediction Intervals for Component Effort

We apply prediction intervals at the team level when making project increment commitments to our customers (where historical data points are components or activities) and at the project level when estimating whole projects (where historical data points are past projects).

2.4.4 Process Performance Models

CMMI v2.0 defines a Process Performance Model as, "A predictive analytical tool that identifies the controllable factors and describes the relationships between measurable attributes of one or more processes, subprocesses, or process element, or work products" [ISACA 2018]. As discussed in Section 2.4.3, our process performance model for our estimation process is key because, from the business perspective, the model helps us make responsible commitments, enabling us to consistently meet or beat them, which helps us delight our customers.

As illustrated by Figure 12, we start with our standardized, stable process (again, in this case, our estimation process) that is "under control" (Section 2.4.6), where we understand the process's capability and variation. This is the "historical process performance data." The historical effort estimates and historical actual effort results are the "measurable attributes" of the process. We apply the "new process inputs" (the raw estimates for the new work being estimated) to the model to obtain a "statistically predictable range of process outputs" from those inputs (the "Yk" adjusted estimates, and the LPI/UPI). We use this tool to perform "what if" analyses by changing the "variable control parameter" (the confidence interval) to make business decisions regarding how much risk we are willing to absorb in the circumstance and determine the resulting estimates.

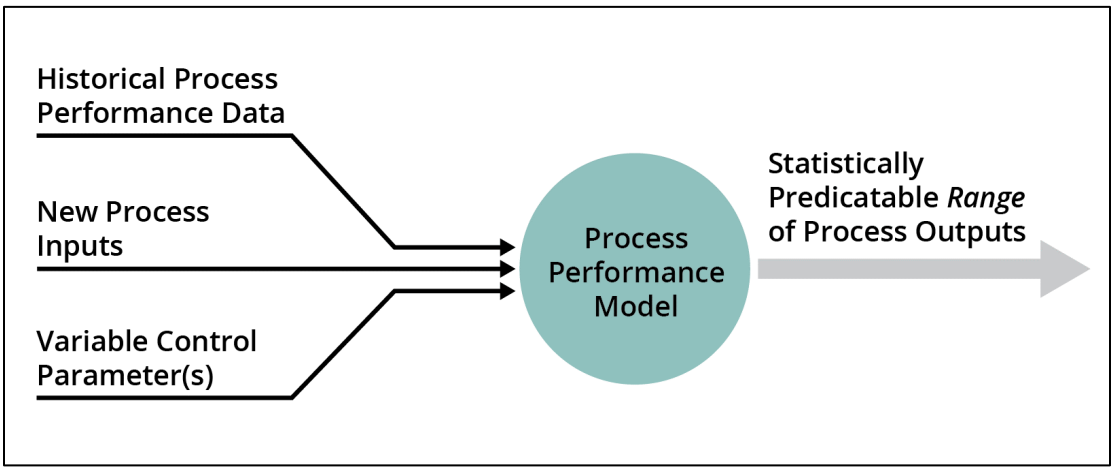

Figure 12: Process Performance Model

As discussed in Section 2.4.3, we are able to apply prediction intervals at the team level when making project increment commitments to our customers (where historical data points are components or activities) and at the project level when estimating whole projects (where historical data points are past projects). Figure 13 depicts how we apply our process performance model at the team level with components.

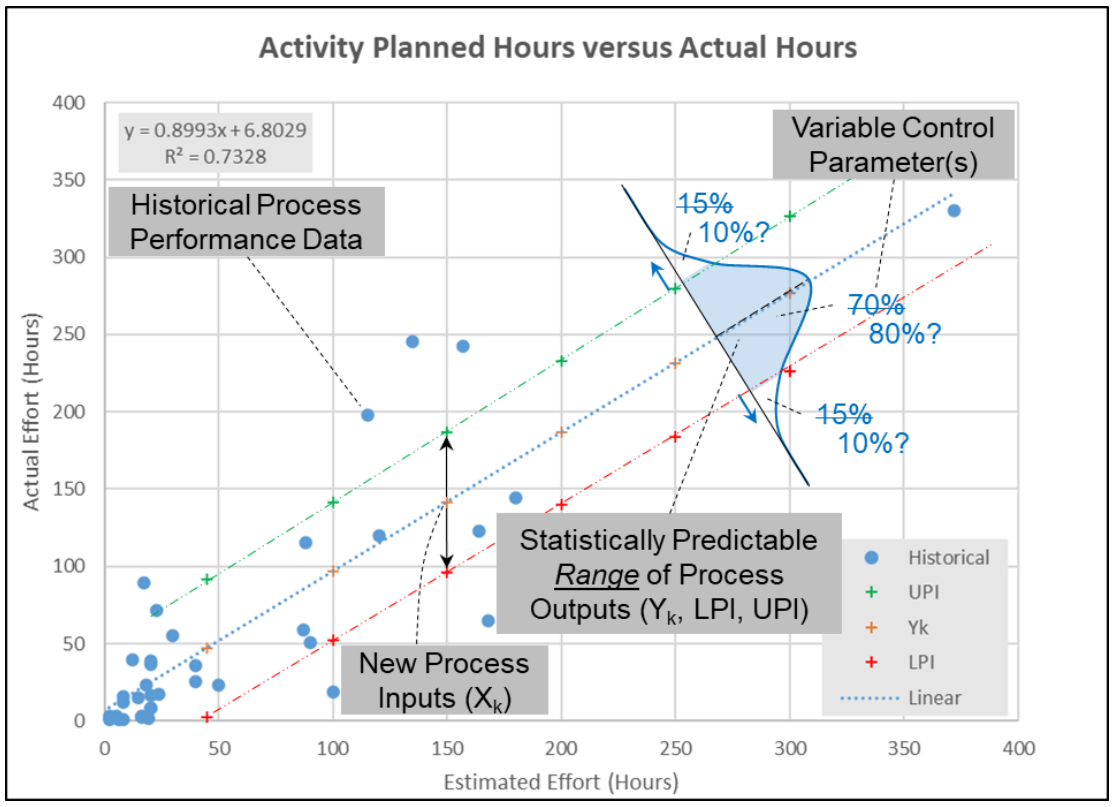

Figure 13: Estimation Process Performance Model Using Historical Data for Components

2.4.5 Histograms

ISHPI uses histograms to analyze data, to help understand and visualize the distributions of data sets, and to understand their similarities and differences. For example, we use them to

- analyze "before" and "after" data (e.g., to analyze the impact of a process change). These data have been helpful in conjunction with testing for statistical significance of a process change (discussed in Section 2.4.7).
- understand the extent to which historical data is similar to new work that we are estimating in terms of its distribution

For example, in Figure 14, a project team was estimating activities for an upcoming project increment. It intended to apply linear regression to correct for bias in its estimates. However, because regression can have a greater impact on small activities than large ones (or vice versa), it was important to understand whether the team's historical data contained a similar proportion of small/large activities as the set of new activities being estimated. We concluded that the distribution was extremely similar and proceeded to apply linear regression on their data.

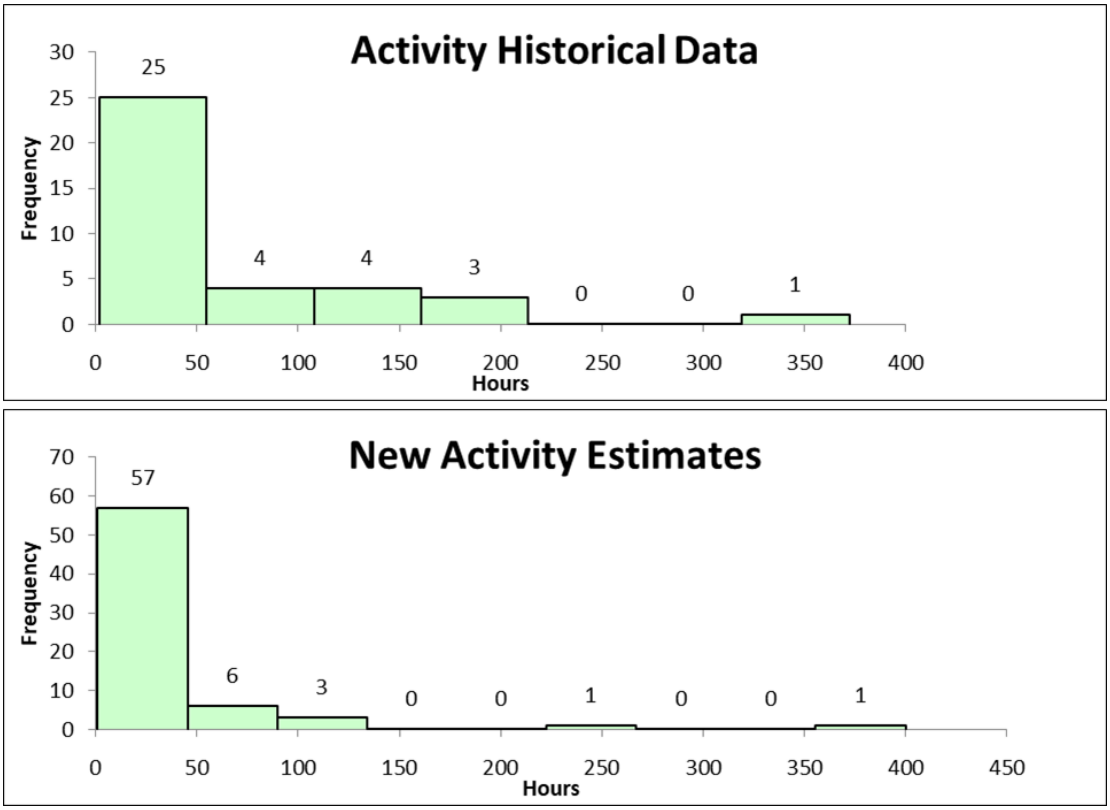

Figure 14: Activity Effort Estimates Analysis

2.4.6 Control Charts

ISHPI uses control charts to monitor the performance of our most important processes to ensure that they are behaving in a natural, predictable way—that is, that they remain "under control." Specifically, we use control charts to understand the process's *capability* (i.e., the range of expected results—the mean and variation), and to understand how the process's capability has evolved over time.

Similar to a run chart, a control chart is simply the sequential plot of a measurable attribute of a process. There are different types of control charts, each of which is useful for analyzing different types of data. We typically use a simple "X Chart," which plots the data points themselves and shows the following (Figure 15):

- the mean of the data points
- boundaries at one standard deviation above and below the mean (i.e., +/- 1σ) defining "zone C"
- boundaries at +/- 2σ (bounding "zone B")
- upper control limit (UCL) and lower control limit (LCL), also known as the natural control limits (+/- 3σ, bounding "zone A")

ISHPI uses "SPC for Excel" to generate control charts. By analyzing the resulting plot, we can understand how much variability exists in the process due to random variation and how much is due to unique events and/or individual actions to determine whether the process is in statistical control.

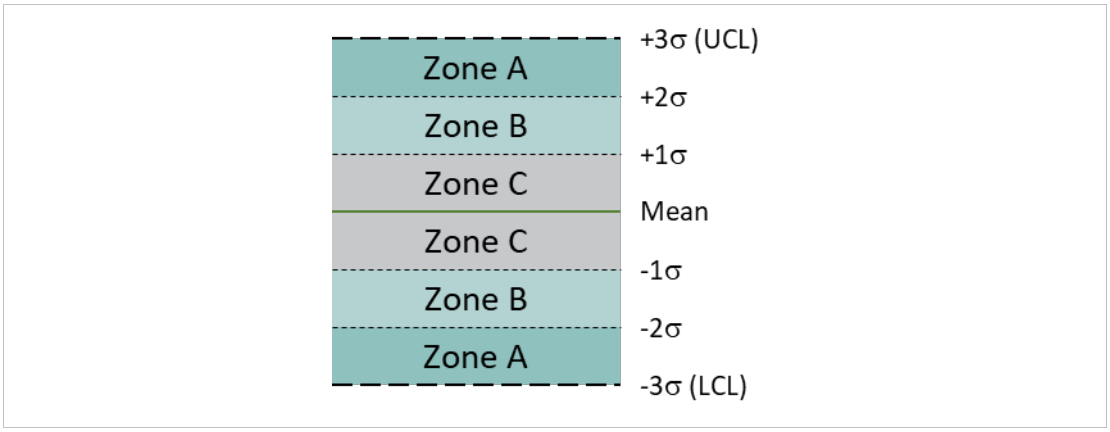

Figure 15: Control Chart Zones

Using the resulting three regions, we analyze the pattern of data in the plotted control chart and determine whether the underlying process is performing "naturally" or if there are any "out of control conditions." We do this analysis by applying "out of control" tests, which identify data points that fall outside the natural limits or form "unlikely" patterns, in which case the process is said to be "out of control." Figure 16 shows the out-of-control tests that we use, as provided by SPC for Excel version 5.

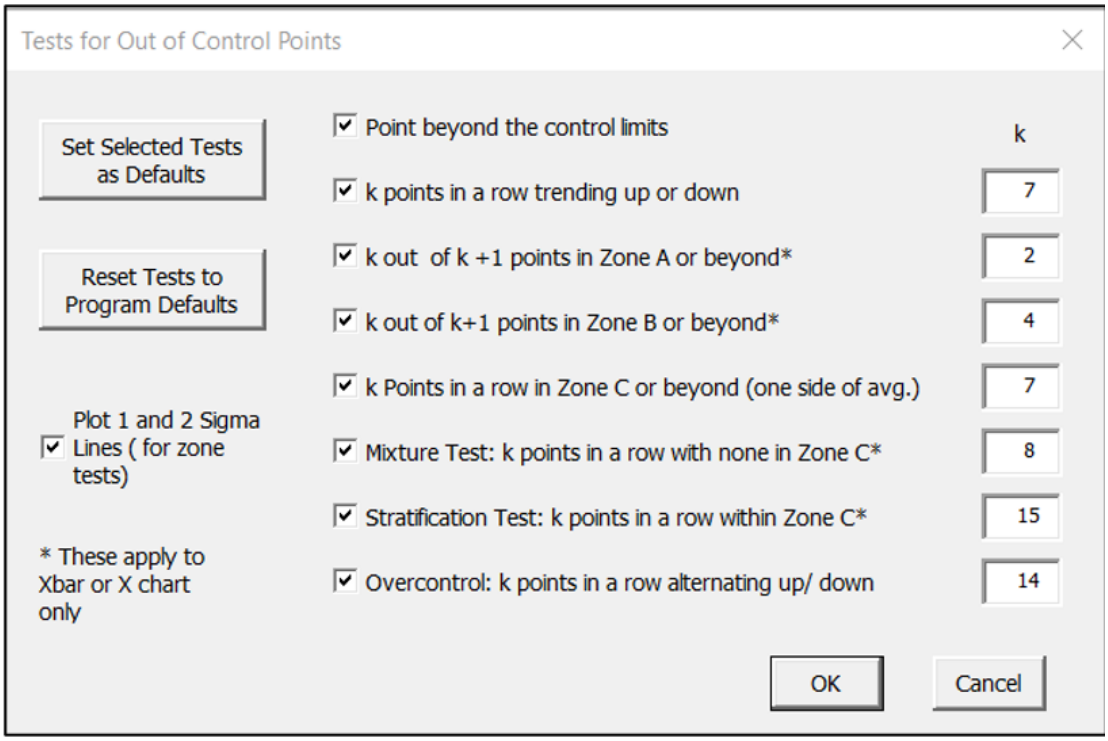

Figure 16: Control Chart "Out of Control" Tests, Provided by SPC for Excel

When using control charts, we are attentive to the following constraints and caveats:

- Data points must be plotted in the sequence the process was performed.
- Typically, at least 15-20 data points must be gathered for a control chart to be considered valid.
- A process found to be "out of control" due to failed tests indicates a "signal"—a special cause. However, this is not necessarily an indication that something is wrong with the process or that something bad has happened. Rather, it indicates that something has changed and that the unusual circumstances need to be investigated.

We provide here a few examples of how *ISHPI* teams use control charts.

Component Estimation: We maintain control charts for component effort estimation error. (An example is shown in Figure 6 in Section 1.5.3.) This is important because we want to make sure team members are estimating their components accurately—that is, aggressively but realistically.

- If we consistently underestimate, we may begin to miss our commitments.
- If we consistently overestimate, our customers may perceive us as being unproductive or inefficient.

Peer Reviews: We maintain control charts for our peer review process because it has such a direct and significant impact on customer satisfaction and our business success. For peer review control charts, we exclude components with a size of fewer than 100 new and changed LOC, since very small components typically have disproportional preparation effort because of the initial overhead

of the review and because they have a small size denominator for defect density, all of which make the process appear out of control.

We maintain preparation rate control charts for peer reviews of components teams develop and maintain (Figure 17) because we want to spend the right amount of time reviewing the work product.

- If we perform peer review preparation too quickly, we might miss defects that would then escape into downstream stages of the development process where they are much more expensive to find and fix.
- If we perform peer review preparation too slowly, it may be inefficient and not yield proportional value.

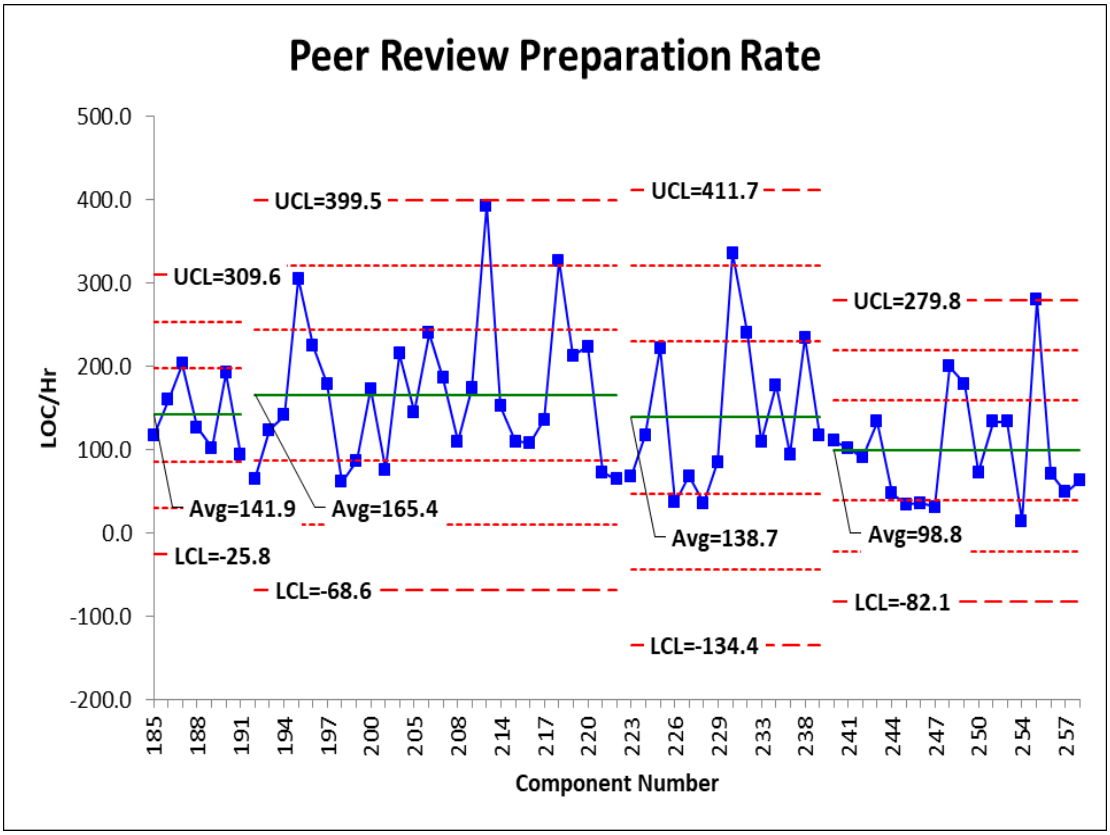

Figure 17: Peer Review Preparation Rate Chart

We maintain defect density control charts for peer reviews of components teams develop and maintain because we want to make sure we are finding the expected number of defects.

- If we find more defects than normal, it may indicate a poor-quality work product that would need further development and remediation before moving into downstream stages of the development process.
- If we find fewer defects than normal, it may indicate an especially high-quality work product (in which case we want to understand the circumstances so that we can reproduce them in the future), or it could indicate a less effective than normal review (e.g., perhaps the reviewer has

insufficient business or technical knowledge to effectively identify defects that may be in the work product).

2.4.7 Tests for Statistical Significance

When *ISHPI* teams implement deliberate, significant process changes to improve their performance, we want to make sure those changes produce the desired effect—quantitatively. We want to make sure that the investments to change our processes provided value and that the improved results did not simply happen by chance. To accomplish this, *ISHPI* uses the "T-Test for Statistical Significance" technique to compare two data sets to determine whether they could statistically be considered part of the same population or whether they are distinct from one another. This involves

- forming a "null hypothesis"
- calculating a "p-value" for the two data sets
- determining the validity of the null hypothesis. A p-value calculated to be less than 0.05 generally indicates that the null hypothesis should be rejected.

The following example illustrates how we have used this technique. On one recent project, routine review of the team's data raised a concern that weekly time being spent in some specific, "ongoing" (i.e., non-earned-value) effort seemed to have increased dramatically. We plotted the weekly data and observed a pronounced increase during the previous six weeks (as compared to the previous period). We plotted the associated control chart (Figure 18) and observed that the previously stable behavior had gone out of control. We quickly concluded that the timing of the increase coincided with a team member change on the project in January 2022. We held a causal analysis session during which we identified several root causes and associated action items related to additional education and mentoring for the new team member. After completing the action items at the beginning of March 2022, we continued to collect and monitor the weekly ongoing effort. We believed that the process had returned to its previously stable state. However, we performed a test for statistical significance to verify that conclusion.

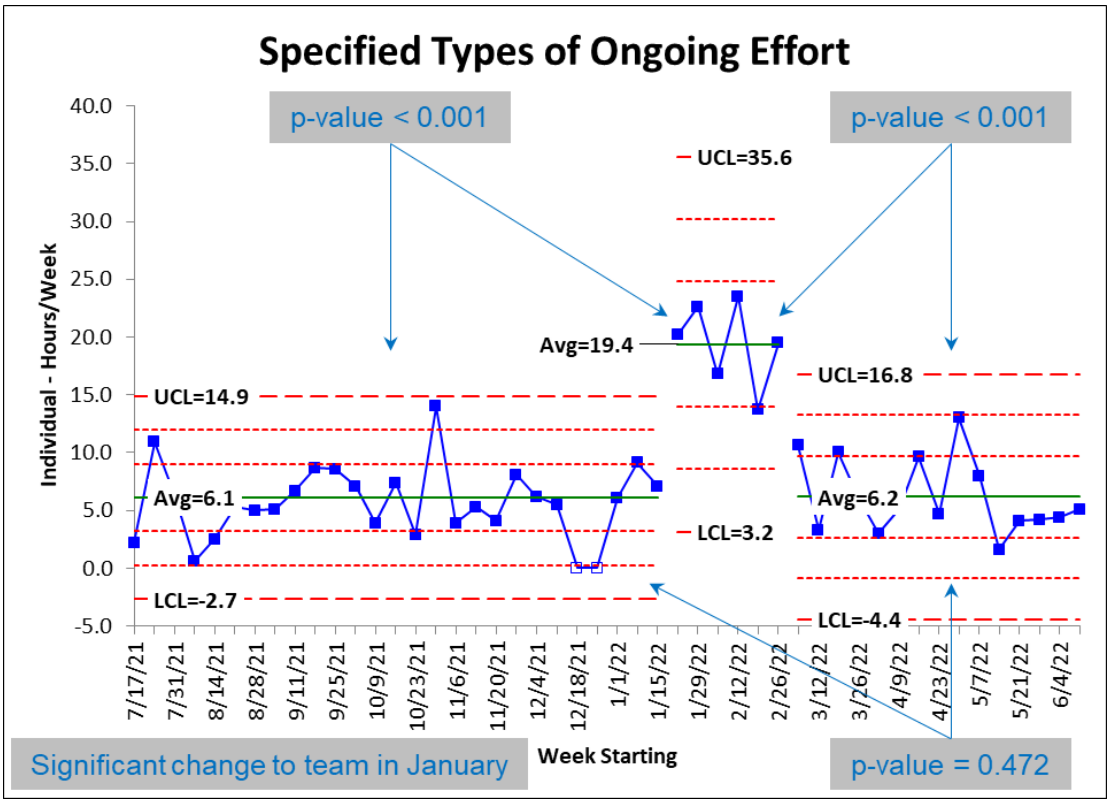

Figure 18: Ongoing Effort Analysis for Statistical Significance

To test the statistical significance of the process changes, we analyzed each of the three groups of ongoing effort data pairwise:

- While comparing data prior to January with data in January and February, we identified the null hypothesis that "the team member change *did not* affect the weekly ongoing effort" and calculated the p-value. The resulting p-value was miniscule (less than 0.001). Since the p-value was far less than the standard 0.05 threshold, we concluded that the null hypothesis should be rejected. That is, the team member change *did* affect the weekly ongoing effort with quite high confidence.
- While comparing data in January and February with data starting in March, we identified the null hypothesis that "the causal analysis action items *did not* affect the weekly ongoing effort" and calculated the p-value. The resulting p-value was again miniscule (less than 0.001). Since the p-value was far less than the standard 0.05 threshold, we concluded that the null hypothesis should be rejected. That is, the causal analysis action items *did* affect the weekly ongoing effort with quite high confidence.
- While comparing data prior to January with data starting in March, we identified the null hypothesis that "the team member change *did not* affect the weekly ongoing effort (after we completed the causal analysis action items)" and we calculated the p-value. The p-value was 0.472. Since the p-value was *not* less than the standard 0.05 threshold, we could *not* reject the null hypothesis. That is, we concluded that the null hypothesis was correct and the process

data sets before January and starting in March were statistically indistinguishable: We had returned the process to its former stable performance.

We recognize that the period in January and February had an extremely small set of data points (only six). Therefore, we understood that the middle portion of the control chart could not be considered valid and that statistical conclusions from such a small data set needed to be taken with extreme caution. However, postponing the causal analysis actions to enable us to collect more data would have been damaging to the project and would not have provided any additional value. Since the differences in performance represented by the three data sets were so pronounced, we were comfortable using the data we had to reach the conclusions and take the actions we did.

2.5 Periodic Analysis of Organization Data

The SEPG conducts an analysis of the organization data prior to the preparation of a Project and Process Status Review (PPSR). The reason for analyzing the process is to draw inferences that can be used to guide decisions and actions to meet *ISHPI*'s business objectives. These decisions and actions lead to updating the process and setting new goals (specification limits) for the organization process performance.

After analyzing the data using the statistical techniques detailed in Section 2.4, the SEPG documents the common causes and the assignable causes for the behavior of a process. If these causes were beneficial, then the process documentation shall be checked and/or updated for concurrency with the practice. If the causes were detrimental, then the process documentation shall be updated to reflect measures to ensure that the results are not repeated.

The process capability baselines and the updated organization goals (specification limits) for process performance are published so the projects, as well as individual practitioners, can compare their performance against an organization benchmark and the specification limits that the management might have set. The organization process capability baselines are published periodically. The preparation of a PPSR is the trigger to publish the new baselines.

3 Significant, Measured, and Sustained Results Using HVD

3.1 Schedule Improvements

The *ISHPI* AIS Division has collected data and metrics on software projects going back to the 1990s and can demonstrate, with actual project performance data, the positive impact of HVD on schedule, effort, and quality of performance. The chart in Figure 19 reflects more than 250 project phases across a variety of technical and functional domains, and it demonstrates schedule improvements after (1) achieving CMM Level 5, (2) adopting TSP and PSP, and (3) HVD. As the chart illustrates, since 2005, *ISHPI* AIS has consistently delivered software within an average deviation of -1.8% of a project's committed schedule.

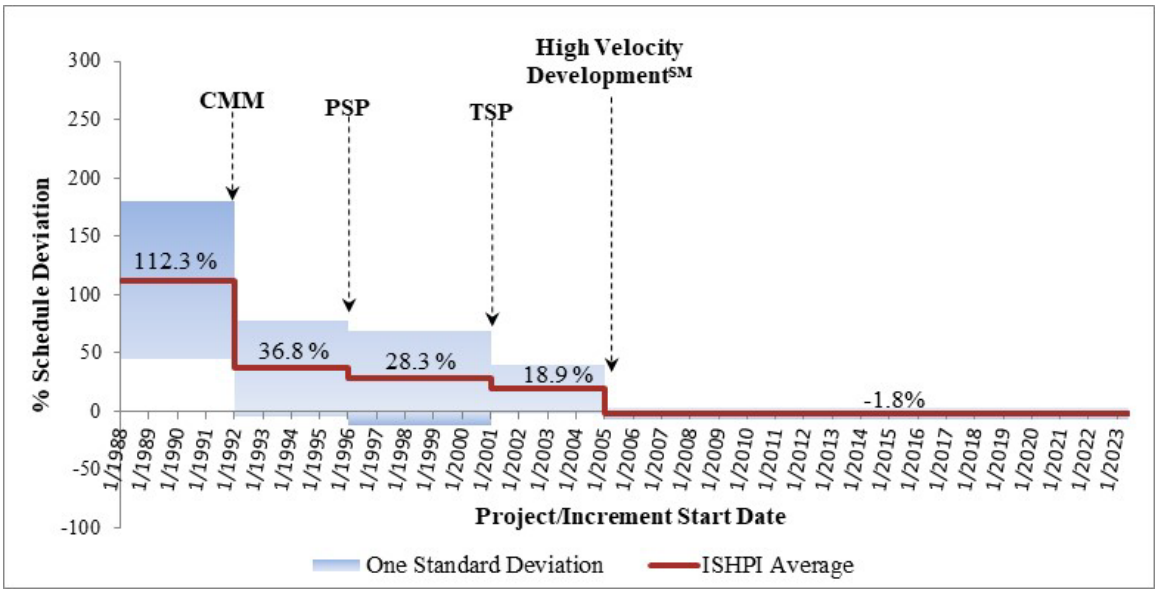

Figure 19: Schedule Deviation Chart for Software Development and Maintenance

3.2 Quality Improvements

Our number one goal is to achieve the highest security and quality possible in the software and other work we produce. To achieve this, *ISHPI* uses a quantitative approach to manage quality. In over 17 years using HVD, the *ISHPI* AIS Division teams have been able to achieve an average of 0.13 defects/KLOC in delivered source code (Figure 20), much less than the industry average, which is upwards of 1 defect/KLOC [Jones 2016]. The result is drastically less rework, leading to an overall cost of quality of 23.7% (Figure 21), which enables excellent schedule performance and yields significant cost savings for customers.

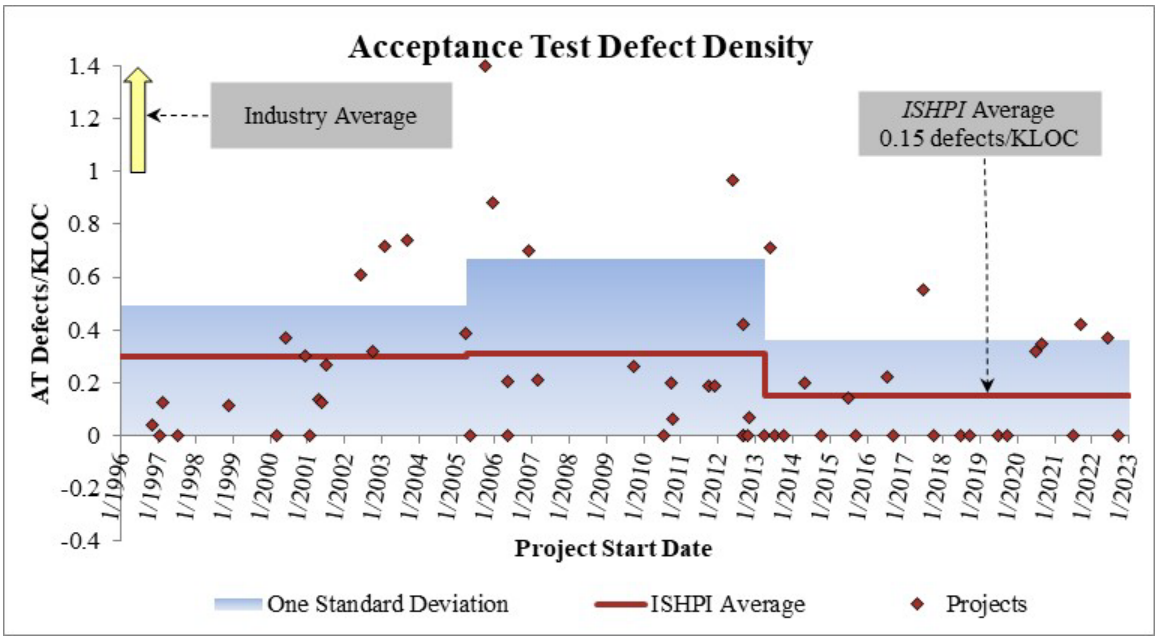

Figure 20: Acceptance Test Defect Density

3.3 Productivity

We have found that *ISHPI* AIS's unique blending of the responsiveness of Agile with the discipline of CMMI ensures the highest quality, lowest lifecycle cost, and greatest customer satisfaction. As a result of these practices, our customers have benefited from shorter schedules and lower costs for development due to minimal rework costs. The result is an average total cost of quality of less than 22% (Figure 21).

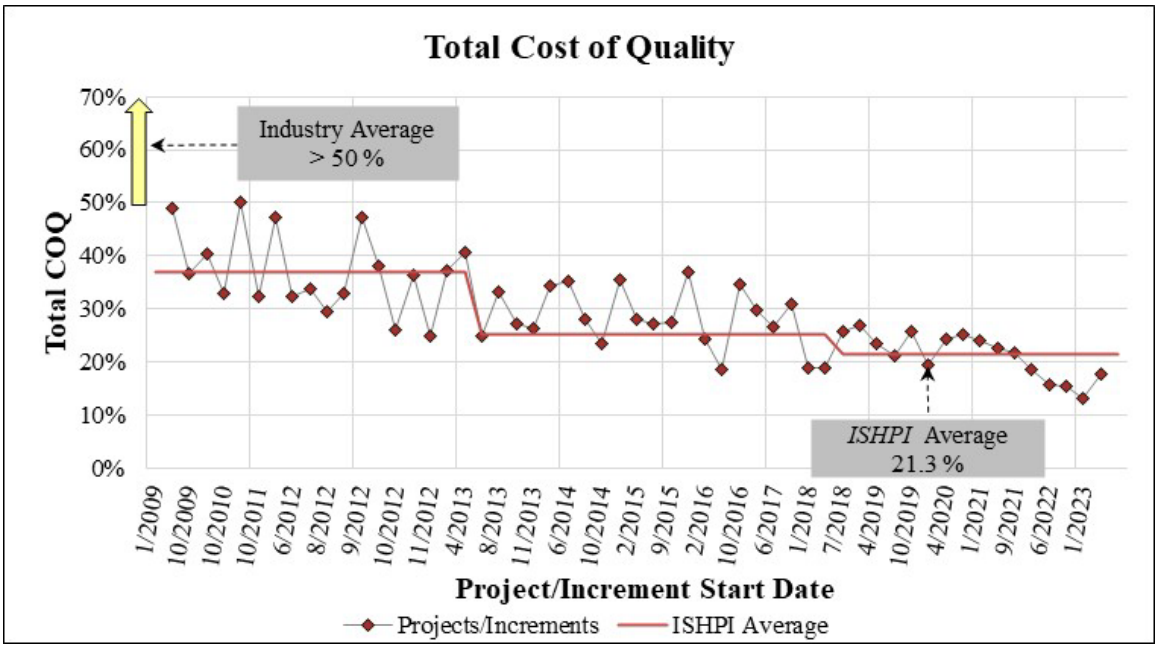

Figure 21: Total Cost of Quality

3.4 First Time Right

The practices and artifacts, finely tuned through continuous improvement, enable our projects to deliver the agreed scope right, the first time, which gives a great experience for our customers. We consistently deliver working software right, the first time, and our customers find few (and often zero) defects. Figure 22 shows that since November 2016, out of 242 support requests triggering a change, only 7 support requests required follow-on effort.

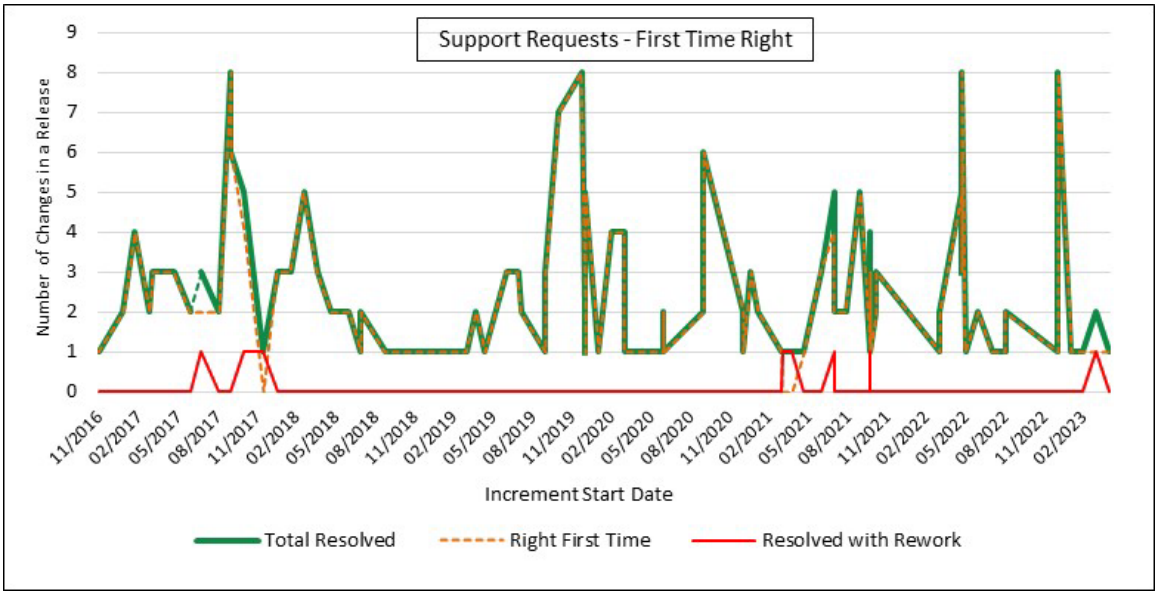

Figure 22: Releases—First Time Right

3.5 Customer Satisfaction

Since 2013, our customers have indicated that we exceeded their needs and/or expectations for value on 97.1% of the engagements. Figure 23 shows that our "exceeded value" customer feedback responses increased from 59.1% to 97.1% since 2005, after adoption of HVD.

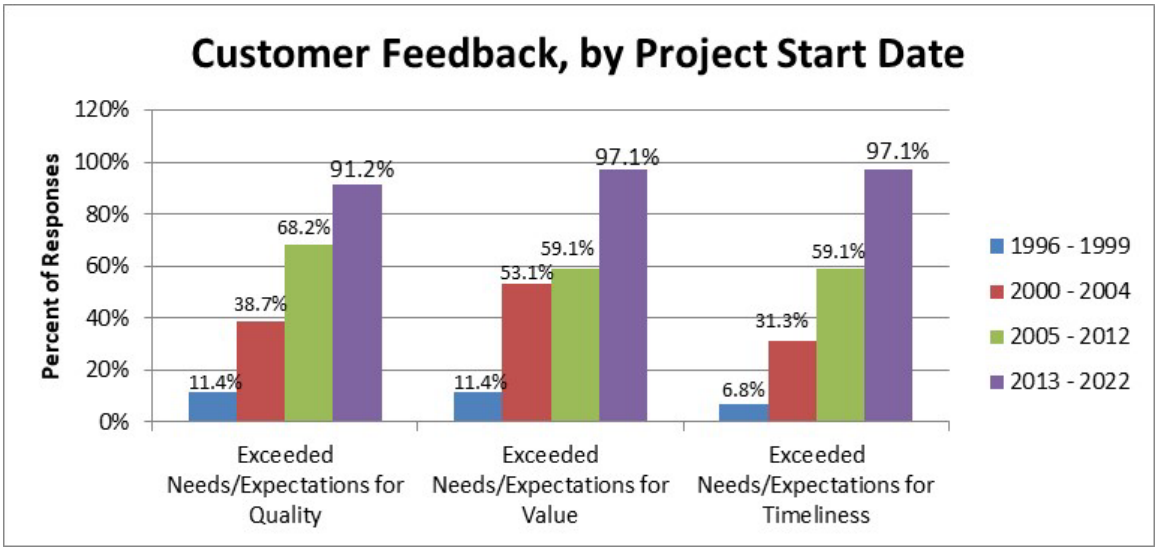

Figure 23: Customer Feedback

3.6 Summary of Management Practices for Producing Secure, Defect-Free Software

Our experience has shown that jelled, self-directed teams consistently collect, analyze, and report the "vital few" data and deliver software that is substantially free of defects and vulnerabilities on predictable cost and schedule. Management bears the responsibility to do the following:

- Staff projects with team members who are formally trained in estimating, planning, tracking, measuring, and managing quality.
- Start each project right with a cohesive team having a shared vision that makes disciplined commitments with the assistance of a coach.
- Support the teams in collecting, analyzing, and reporting product and process data—size, effort, schedule, and quality (defects injected and removed at each step in the process)—providing them the data needed to make decisions and improving continuously through causal analysis and retrospectives.
- Ensure team members are trained to conduct peer reviews of all design and code artifacts to put the highest quality code components into test, striving for 90 to 100% defect-free components with zero cybersecurity vulnerabilities. Ensure the review checklists incorporate specific checks for the types of vulnerabilities identified within the OWASP Top Ten and CWE/SANS Top 25 Most Dangerous Programming Errors.

- Ensure code vulnerability analysis is performed.
- Require teams to report status weekly with precision and accuracy, monitoring planned versus actual numbers of defects throughout the software development lifecycle and enabling the team to proactively manage the quality.

4 Customer Benefits of Agile High Velocity Development℠

4.1 Benefits of HVD

During the past 17-plus years, our teams using *ISHPI*'s HVD practices have achieved results far superior to generally acknowledged industry averages for schedule, cost, and quality of performance, as indicated in Table 8. Our performance is in line with—and often better than—industry performance. Our disciplined approach benefits our customers as shown in Table 7.

Table 7: Features and Benefits of HVD

Features of Our Approach	Benefits
Team launch: We systematically plan team commitments by creating aggressive but realistic plans that are aligned and responsive to customer needs and goals.	We consistently meet our commitments for functionality, cost, and schedule without accumulating technical debt (Figure 19).
We make these commitments using historical data and involving customer stakeholders and all team members, which results in a shared vision and buy-in of the whole team.	We consistently meet our commitments for functionality, cost, and schedule without accumulating technical debt (Figure 19).
We conduct iterative development and delivery with flexible durations. The duration of each iteration is based on customer needs and team strategy.	We maintain the flexibility needed to formulate the most efficient development approach to meet customer needs.
We consistently plan for and rigorously execute quality steps, such as walkthroughs, personal reviews, peer reviews, and tests.	Our delivered products have few security vulnerabilities, which practically eliminates cyber security incidents attributable to poor quality software code (Figure 20).
We practice data-driven monitoring and control.	We gain a quantitative understanding of performance and status.
We act proactively based on risks and the data.	We meet commitments and continuously improve capability and performance.
We practice disciplined engineering.	We consistently deliver working software right the first time, and our customers find few (and often zero) defects (Figure 22).
We consistently meet our commitments for functionality, cost, and schedule without accumulating technical debt.	We achieve high levels of customer satisfaction for quality, value, timeliness (Figure 23).
We practice continuous improvement.	Continuous improvement enables us to perform analyses, provide insights, and offer suggestions that maximize individual and team performance.
We apply our expertise; leverage our ISO 9001:2015 certified quality management system, ISO 20000-1 certified Service Management System, ISO 27001 certified Information Security System; and use our disciplined Capability Maturity Model Integration (CMMI) Maturity Level 5 Agile practices to securely upgrade, enhance, and maintain applications.	These credentials translate to exceptional professionalism, proactive communication, transparency, and visibility with all customer stakeholders with whom we interact, resulting in consistent customer satisfaction.

Features of Our Approach	Benefits
Our approach is intrinsically designed to ensure the security and integrity of the system by building quality and security throughout the lifecycle.	Our approach • reduces tail-end rework, prevents issues, and builds a robust and secure system (Figure 21). • reduces risk to the customer by preventing security vulnerabilities (and associated bad consequences) • results in extremely low total cost of ownership. By delivering secure solutions right the first time, our customers are able to reduce their operating and maintenance (O&M) spending and dedicate more resources to features and strategic initiatives that enhance their ability to fulfill their mission
The *ISHPI* SCOE assists *ISHPI* teams with project launches, retrospectives, and Causal Analysis. It provides support and leverages the power of self-organizing teams.	The SCOE maximizes team productivity in full alignment with customer goals and objectives.

Our productivity and performance improvements after adopting the HVD practices are shown in the column "Productivity/Performance Improvement" of Table 8.

Table 8: Productivity/Performance Improvement Metrics

Performance Category	Definition	Previous Capability (Date Range)	Current Capability (Date Range)	Productivity/ Performance Improvement
Schedule Deviation	Actual schedule divided by planned schedule (by days) times 100 minus 100	18.9% (2001-2004)	-1.8% (2005-2023)	109.5%
Quality: Acceptance Test Defects per KLOC	Defects found during the acceptance test of the work product in 1000 lines of code	0.31 (2005-2012)	0.15 (2013-2023)	51.6%
Total Cost of Quality	Total COQ: appraisal cost, prevention cost, and failure cost • Appraisal cost: effort spent in personal review, peer reviews, and first-time test execution • Prevention cost: effort spent in training, component and milestone retrospectives, and causal analysis • Failure cost: effort spent in analyzing and fixing defects found in reviews and testing	28.1% (2013-2018)	21.3% (2019-2023)	22.2%
Customer Satisfaction: Exceeded expectations for value	Received feedback from customer	59.1% (2005-2012)	97.1% (2013-2023)	64.29%

4.2 Benefits of HVD Practices Based on CMMI DEV Level 5 and CMMI SVC Level 3 Ratings

CMMI Maturity Level 5, the highest possible rating, is characterized as "optimizing" and "high maturity," indicating high performance: It makes us even more predictable, lean, flexible, and cost-effective. *ISHPI*'s CMMI Maturity Level 5 rating is important to us to serve our customers in the following ways:

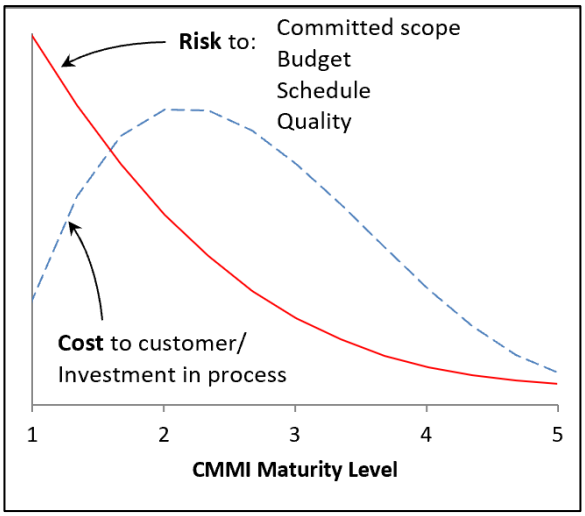

Figure 24: Benefits of CMMI Maturity Level

- Lowest Risk: The practices and artifacts, finely honed through past lessons learned, enable our projects to deliver the agreed scope within budget, on or ahead of schedule, and right the first time, which gives a great experience for our customers.
- Secure Solutions: Disciplined engineering practices practically eliminate cybersecurity incidents attributable to poor-quality software code.
- Lowest Total Cost: By utilizing iPAL, our comprehensive library of reusable and modular objects created through years of PIPs, our customers start the race in the middle, reducing total development cost. By delivering secure solutions right the first time, our customers are able to reduce their O&M spending and dedicate more resources to new features and strategic initiatives that enhance their ability to fulfill their mission.

5 Conclusion and Key Perspectives

The AIS Division's process improvement journey began on January 20, 1992, before "Y2K," when we were still in the age of floppy disks, dot matrix printers, overhead projectors, and transparencies, and when the Internet was mostly still the domain of scientists and universities. The environment today seems to bear little similarity to that time. However, in the face of such significant change over such a long period, a few things have remained constant, and our journey continues.

Throughout our journey, the *ISHPI* AIS Division benefitted from top-down leadership as well as bottom-up involvement in evolving and continuously improving our practices. As then-president of AIS, Girish Seshagiri, established a vision and set a direction, set priorities, and commissioned subject matter expertise and training. He then trusted practitioners to work out the implementation details but required status updates from them—that is, accountability. Although the names and roles have changed, *ISHPI* and the AIS Division follow the same principles.

But, set by Girish's initial direction, what has also remained constant is our persistent focus on

- Quality and security as drivers—We strive to prevent rework to push down costs and increase predictability of effort and schedule because poor quality and the associated rework are inherently expensive and unpredictable.
- Data as a key to understanding—We use data to objectively understand our performance.
- Process improvement as an ongoing activity—Our journey would not have been sustained, and we would not have been able to achieve our results, by relying only on the promise of fantastic new technologies, superstar engineers, expensive new tools, or gifted project managers. Instead, we achieved our results by evolving and continuously improving our process using data.

So, we reaffirm the title of another SEI technical report that our organization wrote decades ago: "Software Process Improvement Works!" [Ferguson 1999]. The fundamental concept of feedback loops and the deceivingly simple "Plan – Do – Check – Act" improvement cycle is indeed a powerful tool that has enormous impact over time. Software process improvement does indeed continue to work.

As we have continued to evolve and mature as an organization, we have gained new insights. We have an innate understanding that our software work is accomplished through the capabilities of individuals, teams, and the organization, and that improvement efforts need to address all those levels. However, over time, we have gained an expanded appreciation that those levels are themselves part of an even larger ecosystem: The "organization" is the *software* organization, and the software organization itself exists within a larger enterprise. Just as the capabilities of a team can enable the best in each individual (and vice versa), so too can the capabilities of the enterprise enable the best in the *ISHPI* AIS Division and each of our other service organizations (and vice versa). Each of the levels interact with and influence one another, and with the right ecosystem, truly amazing things can be accomplished!

This has motivated us to broaden our process improvement efforts and leverage complementary frameworks with a focus on providing value to the overall business. We have now adopted and applied ISO 9001 (Quality Management), ISO/IEC 20000-1 (Information Technology Service Management), CMMI-SVC, ISO/IEC 27001 (Information Security Management Systems), and CMMC (Cybersecurity Maturity Model Certification) frameworks. As a result of these efforts, we quantitatively monitor the performance of enterprise functions such as HR/Recruiting, Business Development, and Finance, among others, and this helps us achieve our enterprise business objectives.

The authors continue to be inspired by Watts Humphrey, the namesake of the Software Quality Award. Particularly notable to us was his ability to draw from other experts—sometimes from the practices of whole other industries—learn from them, build upon them, and apply the underlying concepts to software for the advancement of our industry. He frequently drew from diverse fields, including engineering, psychology, business, political science, and even sports. He built on the ideas of others and used them to innovate. In our own small way, the *ISHPI* AIS Division has accomplished something analogous. We have studied many methods and frameworks (including PSP, TSP, CMMI-DEV, Scrum, Kanban, CMMI-SVC, ISO 9001, ISO 20000-1, ISO 27001, CMMC, and more), adapted them, combined them, and made them our own. The result is an innovative, cohesive process that works for us—our agile High Velocity Development℠ process. We have shown that diverse inputs need not be contradictory choices, but instead complementary building blocks. We have demonstrated that even a small organization that is dedicated to excellence can use all these elements effectively and achieve extraordinary results.

We are on the cusp of a new era in which not only our team members, but the technology itself is learning. We cannot yet perceive how far the earth will shift—and indeed, how far the earth may have *already* shifted. But our decades of experience have taught us that we must keep learning, we must keep sharing, we must keep adapting, and we must keep improving.

Much of the software work we do from day to day seems routine to us. But we are told by experts who occasionally peer in—and we can see for ourselves, when we occasionally rise out of the trenches and survey the landscape—that it is indeed remarkable what our "constancy of purpose" has achieved (and can yet achieve in the future)! We hope that we have honored Watts Humphrey by our practices in some small way, and hope that these practices are, in turn, an inspiration to others.

References/Bibliography

URLs are valid as of the publication date of this report.

[Basili 1994]
Basili, Victor; Caldiera, Gianluigi; & Rombach, H. Dieter. *The Goal Question Metric Approach*. 1994. http://www.cs.umd.edu/~mvz/handouts/gqm.pdf

[Davis 2003]
Davis, Noopur & Mullaney, Julia. *The Team Software Process (TSP) in Practice: A Summary of Recent Results*. CMU/SEI-2003-TR-014. Software Engineering Institute, Carnegie Mellon University. 2003. https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=6675

[Ferguson 1999]
Furguson, Pat; Leman, Gloria; Perini, Prasad; Renner, Susan; & Seshagiri, Girish. *Software Process Improvement Works!* CMU/SEI-99-TR-027. Software Engineering Institute, Carnegie Mellon University. 1999.
https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=13531

[Humphrey 1995a]
Humphrey, Watts S. Introducing the personal software process. *Annals of Software Engineering*. Volume 1. Number 1. December 1, 1995. Pages 311-325. https://link.springer.com/content/pdf/10.1007/BF02249055.pdf

[Humphrey 1995b]
Humphrey, Watts S. *A Discipline for Software Engineering*. Addison-Wesley Publishing Company, Inc. 1995.

[ISACA 2018]
ISACA. Capability Maturity Model Integration (CMMI). *ISACA*. 2018.

[Jones 2016]
Jones, Capers. Achieving Software Excellence. *CrossTalk*. Volume 27. Number 4. July-August. 2014. Pages 19-25. https://apps.dtic.mil/sti/pdfs/ADA604512.pdf

[McAndrews 2000]
McAndrews, Donald R. *The Team Software Process (TSP): An Overview and Preliminary Results of Using Disciplined Practices*. CMU/SEI-2000-TR-015. Software Engineering Institute, Carnegie Mellon University. 2000. http://www.sei.cmu.edu/reports/00tr015.pdf

[Perini 2016]
Perini, Barti; Shook, Stephen; & Seshagiri, Girish. *Reducing Software Vulnerabilities – The Number One Goal for Every Software Development Organization, Team, and Individual*. *NIST Software Measures and Metrics to Reduce Security Vulnerabilities. ISHPI* Technical Report. *ISHPI* Information Technologies, Inc. July 22, 2016. https://ishpi.net/wp-content/uploads/2018/08/SwMM-RSV-Technical-Report.pdf

[Seshagiri 2014]
Seshagiri, Girish. If It Passes Test, It Must Be OK: Common Misconceptions and The Immutable Laws of Software. *CrossTalk.* Volume 27. Number 3. January 2014. Pages 31-25.

[Seshagiri 2015]
Seshagiri, Girish. Performance Metrics That Matter: Eliminating Surprises in Agile Projects. Presented at Software Solutions Conference (SSC) 2015. November 16-18, 2015. https://resources.sei.cmu.edu/library/asset-view.cfm?assetID=447391

[Shook 2020]
Shook, Stephen. Results of Applying Methods for Software Excellence – The Long View. *Software Excellence Alliance*. June 11, 2020. https://softwareexcellencealliance.org/results-of-applying-methods-for-software-excellence-the-long-view/

[Shook 2023]
Shook, Stephen. Application of Statistical and Other Quantitative Techniques in Software. *Software Excellence Alliance*. February 8, 2023. https://softwareexcellencealliance.org/application-of-statistical-and-other-quantitative-techniques-in-software/

[Shull 2013]
Shull, Forrest. A Lifetime Guarantee. *IEEE Software.* Volume 30. Number 6. November 2013. Pages 4-8. https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6648577

[Standish Group 2013]
The Standish Group International. Chaos Manifesto 2013: Think Big, Act Small. The Standish Group International. 2013.
https://www.immagic.com/eLibrary/ARCHIVES/GENERAL/GENREF/S130301C.pdf

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
(Leave Blank)	July 2023	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Software Excellence Through the Agile High Velocity Development℠ Process	FA8702-15-D-0002

6. AUTHOR(S)

Barti K. Perini & Stephen M. Shook

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213	CMU/SEI-2023-TR-002

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100	n/a

11. SUPPLEMENTARY NOTES

12A DISTRIBUTION/AVAILABILITY STATEMENT	12B DISTRIBUTION CODE
Unclassified/Unlimited, DTIC, NTIS	

13. ABSTRACT (MAXIMUM 200 WORDS)

The Advanced Information Services Division of Ishpi Information Technologies, Inc. (DBA *ISHPI*) performs all aspects of the software development lifecycle using its High Velocity Development℠ (HVD) process. We have studied many methods and frameworks (including Personal Software Process, Team Software Process, CMMI for Development, Scrum, Kanban, CMMI for Services, ISO 9001 [Quality Management], ISO 20000-1 [Information Technology Service Management], ISO 27001 [Information Security Management Systems], Cybersecurity Maturity Model Certification, and more), adapted them, combined them, and made them our own. The result is an innovative, cohesive process that works for us—our agile HVD process. We have shown that diverse inputs need not be contradictory choices, but instead complementary building blocks. By evolving, implementing, and utilizing the HVD practices, AIS Division teams have achieved significant improvement in productivity and performance. *ISHPI*'s customers have benefited from shorter schedules, lower costs for development due to minimal rework costs, lower costs for maintenance, and an overall positive experience during each project.

14. SUBJECT TERMS	15. NUMBER OF PAGES
CMMI, CMM, software process improvement, Agile	62

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18 298-102