Outline of PSP Lectures

- Introduction
  - Some background on PSP and How it compares to CMM(I)
- The PSP model
  - PSP 0 - Baseline Personal Process
  - PSP 1 - Personal Planning Process
  - PSP 2 - Personal Quality Management
  - PSP 3 - Cyclic Process
- PSP in Industry
  - Advent case study
  - AIS results

Watt’s books

- A Discipline for Software Engineering, Watts S. Humphrey, 1995
  - The main reference text
  - Some diagrams and exercises were derived from this book.
  - The main text book in abridged format
  - Some diagrams and exercises were derived from this book.
  - Available on module web page
  - You should read this!!
  - An overview of process improvement at each level of the CMM
- Making Process Improvement Personal, Watts S. Humphrey
  - A brief summary a personal software process.

Introduction

- The PSP is a comprehensive methodology that is normally taught as a either an intensive industry course or as graduate or a final year undergraduate course.
- The main objective is to provide a framework for a personal software process
  - Not to teach the specifics of individual process areas.
- The basic message is:
  - As software professionals you should know your own performance.
  - You should measure, track, and analyze your work.
  - You should learn from your performance variations.
  - You should incorporate these lessons in your personal practices

What is PSP?

- A personal process for developing software
  - defined steps
  - forms
  - standards
- A measurement and analyses framework to help you characterize your process
- A defined procedure to help you improve your performance (time, quality)
- Formally definition in web notes

PSP Principles 1

- The PSP is based on planning and quality principles:
  - Every engineer is different; to be most effective, engineers must plan their work and they must base their plans on their own personal data.
  - To consistently improve their performance, engineers must personally use well-defined and measured processes.
  - To produce quality products, engineers must feel personally responsible for the quality of their products. Superior products are not produced by mistake; engineers must strive to do quality work.
  - It costs less to find and fix defects earlier in a process than later.
  - It is more efficient to prevent defects than to find and fix them.
  - The right way is always the fastest and cheapest way to do a job.

PSP Principles 2

- To do a software engineering job in the right way, engineers must plan their work before committing to or starting on a job, and they must use a defined process to plan the work.
- To understand their personal performance, they must measure the time that they spend on each job step, the defects that they inject and remove, and the sizes of the products they produce.
- To consistently produce quality products, engineers must plan, measure, and track product quality, and they must focus on quality from the beginning of a job.
- Finally, they must analyse the results of each job and use these findings to improve their personal processes.
PSP Subgoals

- Clean compile on first compile
- Successful execution of all test data on first run
- Why?
  - faster (less of your time spent)
  - Better (testing often weak)
- With Stable PSP Usage
  - You can
    - estimate and plan your work
    - meet your commitments
    - resist unreasonable commitment pressures
  - You will also
    - understand your ability
    - be better able to improve
    - know if you are improving

Cost / Benefits of a PSP

- The personal costs of a PSP are as follows.
  - The time required to learn and use it.
  - The emotional cost of maintaining the needed discipline.
  - The potential risk to your ego.
- The benefits of a PSP are as follows.
  - The insight you gain into your talents and abilities.
  - The stimulation of an almost unlimited stream of ideas.
  - The framework it provides for personal improvement.
  - The degree of control you gain over your work.
  - The feeling of pride and accomplishment.
  - An improved basis for effective teamwork.
  - The conviction to do the job the way you know you should.

How much does PSP cost?

- PSP is "phenomenally" expensive
- 2003 Costs...
  - (US) $25,784 per person for PSP training at SEI.
  - (US) $18,984 per person for outsourced PSP training.
  - (US) $14,046 per person for instructor training/licensing.
  - 40 hours per person for PSP instructor training/licensing.
  - 160 hours per person of PSP training (classroom and homework).

The CMM(I) and the PSP

- The PSP was developed with the following approach.
  - Identify those large scale software methods that can be used by individuals.
  - Structure these methods so they can be gradually introduced.
  - The PSP has a maturity framework much like that of the CMM(I), with some key process areas removed, since they can not be practiced by individuals.
  - Software subcontract management and inter-group coordination.
  - Individuals cannot practice this.
  - Requirements management and software configuration management.
  - While individuals can practice this, there implications are better demonstrated in a team environment.
  - Software quality assurance and training.
  - These relate to broader organizational issues.

The PSP Structure (1)
The PSP Structure (2)

- PSP methods are introduced in a series of seven process versions
  - These versions are labeled PSP0 through PSP3, and each version has a similar set of logs, forms, scripts, and standards.
- The process scripts define the steps for each part of the process, the logs and forms provide templates for recording and storing data, and the standards guide the engineers as they do the work.

![Diagram of the PSP Structure](image1)

PSP Process Flow

Logical Structure of PSP

- Requirements
- Code Design
- Test Design
- Code
- Test
- Test Reports
- Logs
- Project Plan Summary
- Finished Product
- Plan Data
- Actual Data
- Plan and actual project and process data

![Diagram of PSP Process Flow](image2)

PSP0 – Baseline Personal Process

- Establish a baseline for measuring progress and to define a foundation on which to improve.
  - The PSP0 process provides:
    - A convenient structure for doing small-scale tasks.
    - A framework for measuring these tasks.
    - A foundation for process improvement.
- The tasks introduced include the following:
  - Define current process (PSP0).
  - Time recording (PSP0).
  - Defect recording (PSP0).
  - Defect type standard (PSP0).
  - Code standard (PSP0.1).
  - Size measurement (PSP0.1).
  - Process improvement proposal or PIP (PSP0.1)

![Diagram of PSP0 Process](image3)

PSP0 Current Process

- Identify your current software development process. If you do not have a regular process, use the following:
  - Design, code, compile, test.
- The process includes the following tasks:
  - Planning: Produce a plan to do the work.
  - Development: The actual software development.
  - Postmortem: Comparison of actual performance with your plan.

![Diagram of PSP0 Current Process](image4)

PSP0 - Time Recording

- A time recording log is used to clock the time you spend in each part of the PSP process.
  - The objective is to determine where you spend the bulk of your time.
  - Use sufficient granularity for data to be meaningful - probably minutes.
- The time log should include the following:
  - Date of entry.
  - Start time.
  - Stop time.
  - Estimate time of interruptions for this entry.
  - Delta time - the time you actually work for this entry.
  - Phase being worked on.
  - Pertinent comments.
  - C (completed)
  - U (units)

![Table of Time Recording Log Example](image5)
PSP0 - Defect Recording

- A defect recording log is used to hold data on each defect found and corrected.
- The objective is to determine where you spend the bulk of your time.
- Use sufficient granularity for data to be meaningful – probably minutes.
- The defect log should include the following.
  - Date defect was found.
  - Defect number.
  - Defect type (documentation, syntax, assignment, interface, etc.).
  - Phase where defect was injected.
  - Phase where defect was removed.
  - Time it took to fix the defect.
  - If injected while fixing another defect, that defect’s number.
  - A succinct description of the defect

PSP0 - Defect Type Standard

<table>
<thead>
<tr>
<th>Type Number</th>
<th>Type Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>Documentation</td>
<td>comments, messages</td>
</tr>
<tr>
<td>20</td>
<td>Syntax</td>
<td>spacing, punctuation, type, instruction formats</td>
</tr>
<tr>
<td>30</td>
<td>Build, package</td>
<td>change management, library, version control</td>
</tr>
<tr>
<td>40</td>
<td>Assignment</td>
<td>declaration, duplicate names, scope, limits</td>
</tr>
<tr>
<td>50</td>
<td>Interface</td>
<td>procedure calls and inferences, I/O, user formats</td>
</tr>
<tr>
<td>60</td>
<td>Checking</td>
<td>error messages, inadequate checks</td>
</tr>
<tr>
<td>70</td>
<td>Debug</td>
<td>structure, syntax</td>
</tr>
<tr>
<td>80</td>
<td>Function</td>
<td>logic, pointers, index, recursion, computation, function defects</td>
</tr>
<tr>
<td>90</td>
<td>System</td>
<td>configuration, timing, memory</td>
</tr>
<tr>
<td>100</td>
<td>Environment</td>
<td>design, compile, test, other support system problems</td>
</tr>
</tbody>
</table>

PSP0 - Project Plan Summary

- The project plan summary holds the estimated and actual project data.
- The summary should include the following.
  - Your original estimate of the LOC you expect to develop.
  - The actual LOC you developed.
  - Your original estimate of the time required for each phase.
  - The actual time required for each phase.
  - The total number and the percent of defects injected in each phase.
  - The total number and the percent of defects removed in each phase.

PSP0.1 - Coding Standard

- Develop a coding standard for the following items.
  - Headers.
  - Use/Reuse.
  - Identifiers.
  - Comments.
  - Major Sections.
  - Blank Spaces.
  - Indenting.
  - Capitalization.
- The standard should include definitions and examples

PSP0.1 - Size Measurement

- During software planning process, include an estimate of job size as a function of lines of code (LOC).
- Develop of standard that deals with how to count the following.
  - Modified or deleted lines.
  - Comments or blank lines.
  - Lines with multiple statements.
  - Empty or null statements such as continue, break, semicolon, ....
  - Included files (count once or multiple times).
  - Inline functions or macro expansion.
  - Declarations.
  - Labels.
  - Symbols such as (), begin/end, then, else, case, ....

PSP0.1 - Process Improvement Proposal

- The process improvement proposal (PIP) provides a record of process problems and improvement ideas.
- The PIP should contain the following items.
  - Describe problems encountered on this project.
    - Number each problem.
    - Describe any difficulties.
    - Describe the impact problem had on the product or process.
  - Describe proposals for process improvement.
    - Number each proposal.
    - Identify the specific process element that is affected.
    - Where appropriate, correlate proposal to problems.
    - Prioritize the proposal and defend why it is important.
  - Add overall comments about the project.
    - Lessons learned.
    - Conditions you need to remember to determine why the process worked particularly well or poorly.
PSP0 - Process Script

- Planning.
  - Produce or obtain a requirements statement.
  - Estimate the total new and changes LOC required (PSP0.1).
  - Estimate the required development time.
  - Enter initial project data in the project plan summary.
  - Enter initial data in the time recording log.
- Development.
  - Design, Implement, Compile, Test (PSP0.1).
  - Collect time recording log data.
- Postmortem.
  - Complete the project plan summary with actual time, defect, and size data.
  - Complete the PIP (PSP0.1).

PSP1 - Personal Planning Process

- PSP1 introduces software size estimating and test reporting to the PSP.
  - In addition, PSP1.1 enhances PSP1 to include resource and schedule estimating.
- The PSP1 process is intended to establish an orderly and repeatable procedure for developing software using size, resource, and schedule estimating.
- This estimation process will become progressively more accurate as more data is gathered from various projects.
- The tasks include all of the PSP0 and PSP0.1 tasks plus the following.
  - Size estimating (PSP1).
  - Test reporting (PSP1).
  - Task planning (PSP1.1).
  - Schedule planning (PSP1.1).

PSP1 - Size Estimating

- Any of the following approaches can be used to estimate LOC. You should have the actual size data on a number of previously developed projects to establish a baseline initial estimates.
- PROBE (PRoxy-Based Estimating) Method
  - Described in Discipline for Software Engineering.
- Function Points
- COCOMO Model (Constructive Cost Model)
  - This model is used to make schedule and resource estimates from LOC estimates.

PSP1 - Test Report

- The test report is used to maintain a record of the tests run and results obtained.
  - They should be detailed enough so you can later repeat the same tests and get the same results.
  - The report should include the following.
    - Test name and number.
    - Test objective.
    - Test description.
    - Any special configurations or timing conditions.
    - Expected results.
    - Actual results.

PSP1.1 - Task Planning

- Task planning involves estimating the development time and completion data for each project task.
  - It also provides a basis for tracking schedule progress.
  - The task plan should contain the following.
    - Task number and name.
    - Planned hours by task, by the week, and for the project.
    - Actual hours by task, by the week, and for the project.

PSP1.1 - Schedule Planning

- The schedule is used to record actual hours expended by calendar period.
  - It is used to relate planned tasks to the calendar schedule.
  - The following schedule uses weeks. For small projects, it might be better to track effort by the day.
  - The schedule should contain the following.
    - Week number for each week, typically starting with 1.
    - Calendar date for each week.
    - Planned hours you expect to work on the project that week.
    - Cumulative planned hours.
    - Actual hours you spent working on the project that week.
    - Cumulative actual hours.
**PSP1 - Process Script**

- **Planning.**
  - Produce or obtain a requirements statement.
  - Estimate the software size and required development time (PSP1).
  - Complete the task plan (PSP1).
  - Complete the schedule plan (PSP1).
  - Enter initial project data in the project plan summary.
  - Enter initial data in the time recording log.
- **Development.**
  - Design, Implement, Compile, Test.
  - Collect test report data (PSP1).
  - Collect time recording log data.
- **Postmortem.**
  - Complete the project plan summary with actual time, defect, and size data.
  - Complete the PIP.

**PSP2 - Personal Quality Management**

- The PSP2 process introduces design and code reviews and quality measurement and evaluation.
  - These types of reviews will improve the quality of your software more than any other single change you make to your personal software process.
- The PSP2.1 process introduces design completeness criteria and design verification.
  - The tasks include all of the PSP1 and PSP1.1 tasks plus the following.
  - Design reviews (PSP2).
  - Code reviews (PSP2).
  - Design templates (PSP2.1).

**PSP2 - Reviews**

- The main objective of quality reviews is to improve the quality of programs by examining part or all of a software system or its associated documentation.
- Technical reviews or program inspections are similar, except that their main objective is the identification of defects such as code anomalies, logical errors, or non-compliance with standards.
- Reviews have a number of advantages over dynamic testing.
  - They do not require the program to be run.
  - They are a direct measure of defects or of quality attributes.
  - They are considered more cost effective.

**Review before compiling!**

- Takes as long to do review before as after
- Saves compile time
  - Typically 10% to 15% if done without review
  - Typically ~3% if done with review
- Compiles finds errors equally well with or without review
- Reviewers don’t find errors as effectively if they know the code has compiled cleanly.
  - Probably human nature:
    - if we don’t really expect to find things, we don’t look as hard.
    - if we expect to find things and don’t, we probably look harder.

**PSP2 - Code Reviews**

- Check variable and parameter initialization.
  - At program initialization, start of loops, and function entry.
- Calls function call formats.
  - Pointers, parameters, and the use of ‘&’.
- Check name and spelling use.
  - Is it consistent?
  - Is it within the declared scope?
  - Do all structures and classes use ‘.’ and ‘->’ correctly?
- Check strings.
  - Identified by pointers and NULL terminated.
- Verify all files.
  - Properly declared.
  - Properly opened and closed.

**PSP2 - Code Reviews (cont.)**

- Check pointers.
  - Initial to NULL.
  - Deleted only after new.
  - Always deleted after use.
- Check the output format.
  - Line stopping and spacing is proper.
- Verify logical operators.
  - Proper use of logic operators (==, <, >, etc.).
  - Proper use of parenthesis for logic operations.
- Ensure that brackets ‘{}’ are matched.
- Check every LOC for instruction syntax and proper punctuation.
- Ensure the code conforms to the coding standards.
**Code review script**

<table>
<thead>
<tr>
<th>Step</th>
<th>Task</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Check that the following are on load: 1. The requirements statement. 2. The program design. 3. The program source code. 4. The coding standards.</td>
</tr>
<tr>
<td>2.</td>
<td>Verify the problem.</td>
</tr>
<tr>
<td>3.</td>
<td>Verify that the complete program source code is correct.</td>
</tr>
<tr>
<td>4.</td>
<td>Verify that the code review: 1. Checks the code review. 2. Checks the code review. 3. Checks the code review.</td>
</tr>
</tbody>
</table>

**PSP2 - Design Review**

- Ensure that requirements, specification and high-level design are completely covered by the design.
- All specified outputs are produced.
- All needed inputs are furnished.
- All required includes are stated.
- Verify program logic:
  - Sequencing of stacks, lists, recursion, etc.
  - All loops are properly initiated, incremented, and terminated.
- Check all special cases:
  - Ensure proper operation with empty, full, min, max, negative, zero values for all variables.
  - Protect against out-of-limits, overflow, underflow conditions.
  - Ensure “impossible” conditions are impossible.
  - Handle all incorrect input conditions.

**PSP2.1 - Design Templates**

- There are four design templates that provide an orderly framework and format for recording your designs.
- The formats will not tell you how to do the design, but can help you to properly record the design when you are done.
- The templates include the following:
  - Base specification template.
  - State specification template.
  - Logic specification template.

**PSP2.1 - Operational Scenario Template**

- This template holds descriptions of likely operational scenarios to be followed in using the program.
- The template should include the following:
  - Scenario number for the scenario or scenario steps.
  - Identify the users’ likely objective.
  - The source of the scenario action, such as user, program, or system.
  - Describe the action that takes places, such as providing an error message for incorrect input.
  - List any significant comments.

**PSP2.1 - Functional Specification Template**

- This template can be used to describe functions and procedures for function designs or objects for object-oriented designs.
- The template should include the following:
  - The class/function name and the classes from which it inherits.
  - Any parameters or attributes whose values are externally visible or that impact object behavior.
  - Document methods for each object, including prototype, required variables, types, and the operation performed.

**PSP2.1 - State Specification Template**

- This template is used to document the state behavior of the program, subprogram or classes in an object-oriented system.
- The template should include the following:
  - State name.
  - A description of the state.
  - The attributes or variables that characterize the state.
  - For each state
    - List the all possible next states.
    - List the transition conditions for each next state.
    - Give examples of each transition condition.
PSP2.1 - Logic Specification Template

- This template holds the pseudocode logic for each function or program unit.
- The template should contain the following:
  - List all new or unusual includes required by this function.
  - List all unusual or special types.
  - Give the function declaration or prototype.
  - Document any auxiliary information required to understand the function.
  - Give a number or label to each significant logic statement.
  - Document the program logic.
  - Use pseudocode.
  - Use a separate line for each significant function.
  - Use mathematical or common language statements for clarity.
  - Include comments where necessary.

PSP2 - Process Script

- Planning:
  - Produce or obtain a requirements statement.
  - Estimate software size and required development time.
  - Complete the task plan.
  - Complete the schedule plan.
  - Enter initial project data in the project plan summary.
  - Enter initial data in the time recording log.
- Development:
  - Design, Implement, Compile, Test.
  - Add design review and code reviews (PSP2).
  - Use design template where appropriate (PSP2.1).
  - Collect test report data.
  - Collect time recording log data.
- Postmortem:
  - Complete the project plan summary with actual time, defect, and size data.
  - Complete the PIP.

PSP3 - Cyclic Personal Process

- PSP3 combines multiple PSP2.1 processes to support larger scale software development.
- The main objective is to extend the PSP to industrial-sized projects and to cover team project work.
- This strategy focuses on structuring product development into increments suitable for cyclic development.
- The tasks include all of the PSP2 and PSP2.1 tasks plus the following.
  - Cyclic development (PSP3).

PSP3 - Cyclic Development

- When using PSP3, you should plan to implement larger programs in incremental modules of about 100 LOC (or some other appropriate size).
- Along with a project summary, PSP3 will use cycle summaries to track data on:
  - Program size.
  - Time spent in each development phase.
  - Defects injected.
  - Defects removed.
- An issue tracking log may also be used to record issues, problems, and open questions that span multiple cycles.

PSP3 - Cyclic Development Process

PSP3 - Process Script
The PSP Assignments

- When presented as a mini-course, a number of programming assignments may be assigned by the instructor.
- The ground rules for these assignments are as follows.
  - Pay special attention to the PSP tasks.
  - The principal objective of the PSP assignments is not to complete them correctly, but to completely them correctly using the appropriate elements of the PSP.
  - Above all, your work must be correct.
  - It is better to be late than wrong.
  - Normally, if you need more time, you must ask for it.
- PSP assignments are to be individual efforts.
- Technical assistance may be provided by the course instructor or other group members to clarify requirements or PSP tasks only.
- Document in your logs and reports when assistance was provided.

Assignment 1 - Linked List

- Write a program to calculate the mean and standard deviation of a series of \( n \) real numbers using a linked list. The mean is the average of the numbers.
- The formula for standard deviation is ...

\[
\sigma(x) = \sqrt{\frac{\sum (x_i - \bar{x})^2}{n-1}}
\]

Assignment 2 - Count LOC

- Write a program to count the logical lines of code in a program, omitting comments and blank lines. Make sure multiple statements on individual lines are counted multiple times.
- For example, all of the following contain 2 lines code.

\[
\begin{align*}
R(x := 0); & \\
y := x + 5; & \quad x := 10; \quad f(); \quad \text{if}(x == 0) \quad \text{f();}
\end{align*}
\]

- In addition to the total LOC, the program should count totals for each logical unit (i.e. for each function). Also the number of non-executable statements for the total program and each logical unit should be counted.

Issues with the PSP

- Very Bureaucratic, making it more difficult to sustain in industrial setting.
- Significant Training Requirement
  - 2+ Weeks
  - Developed for academic environment
  - difficult to customise for industrial use
- Lacks proper Tool Support for data gathering
- Many companies involved in PSP training have failed to implement it in industrial context
- Significant numbers of engineers who begin PSP training fail to complete course.
- Bureaucratic overhead makes it more difficult to sustain in industrial setting.

Applying the PSP in Industry

- Advent
  - Used PSP in SPI experiment as part of SPIRE project
  - See case study on module web
- AIS
  - Advanced Information Services Inc. an Illinois-based company, with a subsidiary in Madras, India piloted the PSP on a number of projects
  - PSP was introduced to improve scheduling and defect rates.
  - In the following software project, components 1-3 were completed prior to PSP training with components 4-9 completed following PSP training.

Advent

- Advent Software Ltd.
  - Dublin-based company
  - Spire case study
- Implemented PSP
- Results:
  - Much improved development time estimation
  - Reduction in time spent in test
  - Significant reduction in post-delivery defects
  - Capability to find defects earlier.
Advent results

Average Defect Fix Time in Minutes by Phase

<table>
<thead>
<tr>
<th>Phase</th>
<th>Minutes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Review</td>
<td>0</td>
</tr>
<tr>
<td>Compilation</td>
<td>5</td>
</tr>
<tr>
<td>Test</td>
<td>10</td>
</tr>
</tbody>
</table>

AIS Results (1)

AIS Inc. - Project A - Schedule Estimates

<table>
<thead>
<tr>
<th>Weeks</th>
<th>Estimate</th>
<th>Actual</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>2</td>
<td>10</td>
<td>12</td>
</tr>
<tr>
<td>3</td>
<td>15</td>
<td>18</td>
</tr>
<tr>
<td>4</td>
<td>20</td>
<td>22</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>26</td>
</tr>
</tbody>
</table>

Component Number

Pre-PSP  Post-PSP

AIS Results (2)

AIS Inc. - Project A - Schedule Estimating Error

<table>
<thead>
<tr>
<th>Component Number</th>
<th>Estimate Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>3</td>
<td>200</td>
</tr>
<tr>
<td>4</td>
<td>300</td>
</tr>
<tr>
<td>5</td>
<td>400</td>
</tr>
<tr>
<td>6</td>
<td>500</td>
</tr>
<tr>
<td>7</td>
<td>-100</td>
</tr>
<tr>
<td>8</td>
<td>-200</td>
</tr>
<tr>
<td>9</td>
<td>-300</td>
</tr>
</tbody>
</table>

AIS Results (3)

AIS Inc. - Project A - Acceptance Test Quality

<table>
<thead>
<tr>
<th>Location 1 - Non-PSP</th>
<th>Location 1 - PSP</th>
<th>Location 2 - Non-PSP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defects per 1,000 LOC</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

AIS Results (4)

AIS also developed a number of other projects using both PSP and Non-PSP Engineers

<table>
<thead>
<tr>
<th>Project</th>
<th>Non-PSP</th>
<th>Product</th>
<th>Delivery/Mths</th>
<th>Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>3</td>
<td>24 Reqs.</td>
<td>7/5</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>0</td>
<td>19 Reqs.</td>
<td>2/5</td>
<td>12</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>39 Reqs.</td>
<td>15/19</td>
<td>20</td>
</tr>
<tr>
<td>E</td>
<td>2</td>
<td>2,255 LOC</td>
<td>6/6</td>
<td>0</td>
</tr>
<tr>
<td>F</td>
<td>1</td>
<td>1,400 LOC</td>
<td>2/2</td>
<td>0</td>
</tr>
<tr>
<td>G</td>
<td>2</td>
<td>6,196 LOC</td>
<td>2/2</td>
<td>3</td>
</tr>
</tbody>
</table>

AIS Results (5)

The defect rates for these projects can be graphically represented as follows: