Webinar:
Defects: Pay Me Now and Pay Me Later

June 28, 2012
Time

Please note: The audio portion of this webinar is only accessible through the telephone dial-in number that you received in your registration confirmation email.
Joseph Schofield
Independent Consultant, President of the International Function Point Users Group
joescho@joejr.com

Michael Milutis
Director of Marketing
Computer Aid, Inc. (CAI)
Michael_milutis@compaid.com
About Presenter’s Firm

Joe Schofield is a SEI-certified instructor for the Introduction course as well as the Level 2 and Level 3 Practitioner courses. Joe taught these courses for many years before retiring from Sandia National Laboratories as a Distinguished Member of the Technical Staff. Spanning 31 years he led in the development of numerous IS/IT projects and the last thirteen of those years, he was directly responsible for the implementation of the CMMI® for an organization of 400 personnel. In addition he facilitated over 100 teams in the areas of software specification, team building and organizational planning by using lean six sigma and business process reengineering.

Joe serves as the President of the International Function Point Users Group, and is a CFPS, CSMS, CSQA, and a Lean Six Sigma black belt. He has over four dozen published papers, conference presentations and keynotes—including contributions to the books IT Measurement, Certified Function Point Specialist Exam Guide, The Economics of Software Quality, and just released, The IFPUG Guide to IT and Software Measurement. Joe is a frequent presenter in the Software Best Practices Webinar Series sponsored by Computer Aid, Inc. Joe has taught over 75 graduate courses since 1990.
About Computer Aid, Inc. (CAI)

- **CAI** is a global IT outsourcing firm currently managing active engagements with over 100 Fortune 1,000 companies and government agencies around the world.
- **CAI** is a leader in IT Best Practices for legacy support and new development application management.
- **CAI’s** focus is directed toward practical implementations that track and measure the right activities in software activity management.
- **CAI** consistently promises and delivers double digit productivity in its outsourcing and consulting engagements.
- **CAI** makes all of this possible through the use of:
  - Standard processes
  - Management by metrics
  - SLA compliance management
  - Detailed cost, resource, and time tracking
  - Capacity management
  - Standard estimation
  - A unique, metrics based methodology along with a proprietary, real time data repository and management system (**tracer®**).
NOW AVAILABLE!
ONLINE WEBINAR RECORDINGS
ANYTIME ACCESS!

WWW. ITMPI.ORG / LIBRARY
PDU CREDITS FOR THIS WEBINAR

• The Project Management Institute’s ISSIG group has accredited this webinar with PDUs

• Stay tuned! Your PDU code will be displayed at the conclusion of this webinar.
Your logo here
First question: Shouldn’t the title of the presentation be “Pay Me Now OR Pay me Later” – absolutely not! This discussion isn’t about changing the oil in your car!

“Yes” this message could be restated more positively as “save now and save forever.”

Defects increase the cost of development AND increase the cost of support.

Missing requirements (“we’ll handle that in the next release”) increase the likelihood of defect injection or the “we’ll get it right the next time” syndrome.

In some cases, adding “re-work” is as simple as placing a product on a sprint backlog; however, by itself, doesn’t trigger a defect detection.

The pervasiveness, sophistication, and architectural diversity of software further increases the cost of software defects. The title of this presentation might ought to be: “Pay Me a Lot Now, and a Lot More Later.”
## Rationale for the Premise

<table>
<thead>
<tr>
<th>Pay Me Now (during development)</th>
<th>and . . . Pay Me Later (during support)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Defect and rework costs tend to be “hidden” during development as customer changes, scope creep, “testing”, and more recently “refactoring.”</td>
<td>These costs tend to be more visible but regarded as the cost of operations and maintenance.</td>
</tr>
<tr>
<td>These costs tend to be masked as schedule slippages and unexpected cost overruns. (no reason to consider these <em>unexpected</em> given history)</td>
<td>These costs will increase as defect removal occurrences become more frequent. (Watts’ work on inserting more defects more likely during defect removal).</td>
</tr>
<tr>
<td>These costs are discoverable, treatable, and predictable. Measurements of past work, requirements volatility, test results (% in code vs. design and requirements), and latent defect prediction are leading indicators of the existence of these hidden costs.</td>
<td>Requirements elicitation, development, and management likely continue into operations and support and can be measured with volatility metrics.</td>
</tr>
<tr>
<td>These costs can be reduced with peer reviews &amp; inspections, defined processes (as in CMMI-DEV ML3 and higher organizations).</td>
<td>These costs can be predicted from latent defect estimation techniques during development.</td>
</tr>
</tbody>
</table>
# Sample Software Defects: Pervasive and Sophisticated

<table>
<thead>
<tr>
<th>Who</th>
<th>What</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Home Depot</td>
<td>Malicious code</td>
<td>foxnews.com website; retrieved 1/11/2011</td>
</tr>
<tr>
<td>Toyota</td>
<td>Prius brake software</td>
<td>green.autoblog.com; February 9, 2010</td>
</tr>
<tr>
<td>BMW, DaimlerChrysler, Mitsubishi, and Volvo</td>
<td>Engine stalls, non-illuminating gauges, incorrect wiping intervals, and wrong transmission gears</td>
<td>Software Quality, informationweek, 3/15/2004</td>
</tr>
<tr>
<td>Medical staff</td>
<td>Prescription errors</td>
<td>Medication Systems; CIO; June, 2005: 28</td>
</tr>
<tr>
<td>Marriott, Ford, Justice Department, T J Maxx</td>
<td>Data breeches / loss</td>
<td>USA Today; October 25, 2007</td>
</tr>
<tr>
<td>MGM Hotels (5 of 7 on LV strip)</td>
<td>Registration software</td>
<td>Las Vegas Review-Journal; October 24, 2007</td>
</tr>
<tr>
<td>Cancer treatment center - Panama</td>
<td>21 deaths in 40 months from software-related overexposure to gamma rays</td>
<td>Baseline – The Project Management Center, We Did Nothing Wrong, March 4, 2004</td>
</tr>
</tbody>
</table>
# Evidence of Paying Now and Paying Later (general)

<table>
<thead>
<tr>
<th>Cost</th>
<th>Now</th>
<th>Later</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>NIST reports that 80 percent of software development costs are traceable to <em>defect corrections</em>.</td>
<td>X</td>
<td></td>
<td>The Economic Impacts of Inadequate Infrastructure for Software Testing; National Institute of Standards &amp; Technology; US Dept of Commerce; May, 2002</td>
</tr>
<tr>
<td>38 percent of polled organizations have no SQA program.</td>
<td>X</td>
<td>X</td>
<td>CIO, By the Numbers, December 1, 2003, pg 28</td>
</tr>
<tr>
<td>Software defects cost the U.S. $59.6B a year (almost a decade ago).</td>
<td></td>
<td></td>
<td>Informationweek, Behind the Numbers, March 29, 2004; pg 94</td>
</tr>
<tr>
<td>Thirty percent of project effort can be traced to rework.</td>
<td>X</td>
<td>X</td>
<td>Dr. Dobb’s Report; informationweek; July 12, 2010</td>
</tr>
</tbody>
</table>
# Evidence of Paying Now and Paying Later (requirements, design, review)

<table>
<thead>
<tr>
<th>Cost</th>
<th>Now</th>
<th>Later</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements defects that are released can cost 50 – 200 times as much to correct as defects that were corrected close to the point of creation.</td>
<td></td>
<td>$X$</td>
<td>Boehm, Barry W. and Philip N. Papaccio. &quot;Understanding and Controlling Software Costs,&quot; IEEE Transactions on Software Engineering, v. 14, no. 10, October 1988, pp. 1462-1477</td>
</tr>
<tr>
<td>Reworking defective requirements, design, and code typically consumes 40 to 50 percent or more of the total cost of most software projects and is the single largest cost driver.</td>
<td>$X$</td>
<td>$X$</td>
<td>Jones, Capers. Estimating Software Costs, New York: McGraw-Hill, 1998.</td>
</tr>
<tr>
<td>As a rule of thumb, every hour you spend on technical reviews upstream will reduce your total defect repair time from three to ten hours.</td>
<td>$X$</td>
<td></td>
<td>Jones, Capers. Assessment and Control of Software Risks. Englewood Cliffs, N.J.: Yourdon Press, 1994</td>
</tr>
</tbody>
</table>
## Evidence of Paying Now and Paying Later (testing through support)

<table>
<thead>
<tr>
<th>Cost</th>
<th>Now</th>
<th>Later</th>
<th>Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>Testing composed 25 – 50 percent of the software life cycle, and it is perceived of adding no business value.</td>
<td>X</td>
<td>X</td>
<td>Two Reasons Why IT Projects Continue To Fail; Gartner; March 20, 2008</td>
</tr>
<tr>
<td>Only ½ of the defects in a product are removed by testing; this limitation is not a reflection on the testing process.</td>
<td>X</td>
<td>X</td>
<td></td>
</tr>
<tr>
<td>47 percent of companies surveyed reported “higher than expected” maintenance costs associated with their software.</td>
<td>X</td>
<td></td>
<td>Dynamic Markets Limited; August 2007</td>
</tr>
<tr>
<td>Small code changes (often introduced during production “fixes”) are 40 times more likely to introduce new defects than original development work.</td>
<td>X</td>
<td></td>
<td>Watts Humphrey. 1995. A Discipline for Software Engineering. Addison-Wesley</td>
</tr>
<tr>
<td>Customer-reported defects cost an average of $4,200 to “deal with.”</td>
<td>X</td>
<td></td>
<td>The Requirements Payoff; Karl Wiegers; informationweek; July 12, 2010; pg 39 study by Dean Lefingwell, 1997</td>
</tr>
</tbody>
</table>
Causes of Paying Now and Paying Later

- lack of current familiarity by the developer (if being fixed by the same person)
- lack of familiarity by the developer (if being fixed by a different person)
- difficulty in finding all references to a changed variable
- difficulty in finding all variables in a calculation
- reliance on outdated comments in the code
- reliance on outdated documentation about the code
- failure to configuration manage all the products related to a defect repair
- failure to update test cases related to defect removals
- (let’s not forget) failure of management to foster a culture of quality
- failure to incent teams and individuals to perform
- the clustering tendency of defects
- lack of training in formal test techniques
- failure to test software against customer requirements (as opposed to the developer’s interpretation of requirements)
- inability to elicit testable requirements
- reliance on software development approaches not intended for software of high consequence
- a reliance primarily on testing to discover defects
Preventing Defect Escapes  
(pay less now and later)  
(Latent defect estimation)

Place a check mark in the intersecting cells for each defect found by each participant. Count the defects that each engineer found (*Counts* for Engineer A, B, and C). Column A: check and count all the defects found by the engineer who found the most unique defects.  5  
Column B: check and count all of the defects found by all of the other engineers.  4  
Column C: check and count the defects common to columns A and B.  2

The estimated number of defects in the product is AB/C. Round to the nearest integer. $(5 * 4) / 2 = 10$

The number of defects found in the inspection is A+B-C.  $5 + 4 − 2 = 7$

The estimated number of defects remaining is the estimated number of defects in the product minus the number found.  $(AB/C) − (A+B-C).  \quad 10 − 7 = 3$

<table>
<thead>
<tr>
<th>Defect No</th>
<th>Engineer Larry</th>
<th>Engineer Curly</th>
<th>Engineer Moe</th>
<th>“Column A”</th>
<th>“Column B”</th>
<th>“Column C”</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>√</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>4</td>
<td>√</td>
<td>√</td>
<td></td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>5</td>
<td>√</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>6</td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>7</td>
<td></td>
<td></td>
<td>√</td>
<td></td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Counts</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>2</td>
</tr>
</tbody>
</table>

Use team “thresholds” to determine whether or not to repeat the Peer Review.

The capture-recapture method (CRM) has been used for decades by population biologists to accurately determine the number of organisms studied. LaPorte RE, McCarty DJ, Tull ES, Tajima N., Counting birds, bees, and NCDs. Lancet, 1992, 339, 494-5.

See also Introduction to the Team Software Process; Humphrey; 2000; pgs. 345 – 350
Quick Hits on Capture-Recapture Method

- CRM can be used to predict the number of estimated defects remaining in a product. This estimate can then be used to make quantified, data-driven decision on how to proceed with a software product. *IFPUG MetricViews; Winter 2008*

- Used by Jerry Weinberg to estimate remaining typos based on how many his reviewers found. *Beyond Defect Removal: Latent Defect Estimation with Capture Recapture Method; CrossTalk, August 2007*

- Try this: Something Fishy (statistics) from PBS Mathline. [www.pbs.org/mathline](http://www.pbs.org/mathline)


- A more detailed look – Life Cycle-Based Defect Removal with Capture Recapture Methods. *Computer Aid, Inc. webinar, April 22, 2008*

- A quickie history – Capture-Recapture History. [www.pitt.edu/~yuc2/cr/history.htm](http://www.pitt.edu/~yuc2/cr/history.htm)
Where do defects go after they die?
(it depends on if they've been saved or not!)

Defects get defined (organizational glossary)

- **Actual effort**: The measure as expressed in FTEs, to complete a unit of work.
- **Actual reliability**: The measure of stability of a product by the customer; measurable.
- **Actual schedule**: The timeline for completing a unit of work.
- **Actual size**: The size of work product as expressed in Function Points.

Defects get prevented (changes to SILC and project defined process)

- **Prevented**: Changes to SILC and project defined process.

Defects get discovered (PR, Testing, Customer)

- **Discovered**: To identify and remove defects early and efficiently from artifacts in cycle in order to improve the delivered product. Note: Design discussion during the inspection meeting is discouraged. Criticism of is prohibited.

**Entry Criteria**:
- Supporting documentation is available as required.
- Adequate review preparation by the Peer Review Team.

**General Process**:
1. Producer notifies Project Leader when artifact is ready for review.

---

**Metrics Database**

- **Captured/saved**: (recorded)
- **Project Plan Defects per Hour Review**

---

*left unsaved (not recorded) these defects are of no value to the process, and, likely to be repeated*
Tips for Avoiding Paying Now and Paying Later

- Where requirements are understood, elicit them with JAD-like sessions, prototypes, user stories, and early validation. Track requirements volatility.
- Where requirements are less understood similar techniques can be used but validation will be less useful. Tracking requirements will only remind you of how “unstable” requirements are and volatile planning may be.
- Invest in inspections (more formal) and peer reviews (less formal) to reduce development time and cost, and to reduce defects.
- Use CRM as a way to determine the quality of inspections and peer reviews before moving a product “downstream.”
- Develop a culture that values process improvement and product quality.
- Train software engineers in the discipline of software engineering and the science of computer science.
- Pilot “newest fads.” Objectively and quantitatively evaluate prior to widespread adoption.
Questions?
CAI Sponsors

The IT Metrics Productivity Institute:

- Clearinghouse repository of best practices: [WWW.ITMPI.ORG](http://WWW.ITMPI.ORG)
- Weekly educational newsletter: [WWW.ITMPI.ORG](http://WWW.ITMPI.ORG) / SUBSCRIBE
- Weekly webinars hosted by industry leaders: [WWW.ITMPI.ORG](http://WWW.ITMPI.ORG) / WEBINARS
- ACCESS WEBINAR RECORDINGS ANYTIME AT [WWW.ITMPI.ORG](http://WWW.ITMPI.ORG) / LIBRARY
Software Best Practices
Conferences Around the World

2009 Dates and Locations

<table>
<thead>
<tr>
<th>Date</th>
<th>Location</th>
<th>Date</th>
<th>Location</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mar. 10</td>
<td>Orlando, FL</td>
<td>May 14</td>
<td>Tampa, FL</td>
</tr>
<tr>
<td>Mar. 19</td>
<td>Princeton, NJ</td>
<td>May 19</td>
<td>Philadelphia, PA</td>
</tr>
<tr>
<td>Mar. 31</td>
<td>Albany, NY</td>
<td>Sep. 15</td>
<td>Toronto, ON</td>
</tr>
<tr>
<td>Apr. 9</td>
<td>Baton Rouge, LA</td>
<td>Sep. 22</td>
<td>Chicago, IL</td>
</tr>
<tr>
<td>Apr. 21</td>
<td>Detroit, MI</td>
<td>Sep. 29</td>
<td>New York, NY</td>
</tr>
<tr>
<td>Apr. 28</td>
<td>Tallahassee, FL</td>
<td>Oct. 6</td>
<td>Baltimore, MD</td>
</tr>
<tr>
<td>May 5</td>
<td>New York, NY</td>
<td>Oct. 20</td>
<td>Philadelphia, PA</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Nov. 17</td>
<td>Miami, FL</td>
</tr>
</tbody>
</table>
Presenter Name 1

Title
Organization
Email Address

Michael Milutis

Director of Marketing
Computer Aid, Inc. (CAI)
Michael_milutis@compaid.com