Performance Metrics That Matter: Eliminating Surprises in Agile Projects

Girish Seshagiri, Ishpi Information Technologies, Inc.

Software Solutions Conference 2015
November 16–18, 2015
“If We Eliminate the Monthly Status Report What Do We Replace It With?”
Agenda

Software Engineering’s Persistent Problems
Common Misconceptions of Software
Immutable Laws of Software Development
Performance Metrics That Matter
Main Points

We do awesome things in IT. Our **problems persist**.
Status quo is not acceptable with the threat of **cyber attacks**

We need

- to shift our focus to the **individual developer/engineer** trained in quality methods
- to cease dependence on test as the principal defect removal method
- the “**vital few**” **performance metrics** that really matter and help us manage the software work by managing quality
IT Acquisition Failures Land On High-risk List

GAO Report

“Too frequently, federal IT investments fail to be completed or incur cost overruns and schedule slippages while contributing little to mission-related outcomes,”

"Unfortunately, fairly consistently, we find problems with these projects. And these seem to center on a lack of discipline and effective management practices, the need for improvements in project planning, and poor program oversight in governance."
Software Engineering’s Persistent Problems - 1

Exponential rise in cybersecurity vulnerabilities due to defective software

Unacceptable cost, schedule, and quality performance of legacy systems modernization and Enterprise Resource Planning (ERP) projects
Software Engineering’s Persistent Problems - 2

Cost of finding and fixing software bugs (i.e. scrap and rework) the number one cost driver in software projects

Arbitrary and unrealistic schedules leading to a culture of “deliver now, fix later”
Software Engineering’s Persistent Problems - 3

**Inability to scale** software engineering methods even for medium size systems

Lack of understanding of the impact of **variation in individual productivity**

Absence of work place democracy and **joy in work**
The Appetite for Assured Software

The organizational appetite for assured software is driven by the net losses realized from compromised software.

The consumer has been living with nearly 60 years of poorly developed and incompetent software.

Hundreds of millions of dollars are spent annually on post software compromise and incident recovery, lost opportunities and productivity (ask me).

Insecure software represents a pervasive kinetic threat to critical infrastructure and our way of life…..make no mistake about it.

The prudent approach is to take a proactive one. That is, software assurance measures must be a top integration priority in the enterprise cyber security risk management schema.

SWAMP Webinar – Jerry L. Davis, Chief Information Officer, NASA
By the Numbers

Feel my pain. Lack of a good software assurance program is a painful experience

At one time – 127 applications were tested and;

- 81 (64%) contained high vulnerabilities that facilitated exposure of sensitive data or system take over;
- 45 applications (36%) exposed Personally Identifiable Information (PII)

At another time – 50 applications were tested and;

- 41 applications (82%) hosted OWASP top 10 defects
- 5 applications (10%) taken offline due to high risk
- 19 (38%) contained high vulnerabilities that facilitated exposure of sensitive data or system take over
- 12 applications (24%) exposed PII

Source: Shaping Your Approach – the Executive's role in software Assurance, SWAMP Webinar, Jerry L. Davis, Chief Information Officer, NASA
Emerging Cyber Threats Call for a Change in the ‘Deliver Now, Fix Later’ Culture of Software Development

By Girish Seshagiri, CEO of Advanced Information Services Inc. (AIS)

The demand for new and innovative technology solutions has created a software industry laser-focused on speed to market, costs and product functionality. While this may help companies achieve a first-to-market advantage, it has also led to an environment where developers are more focused on meeting unrealistic schedule commitments than producing high-quality software.

necessary to permanently reduce the number of vulnerabilities found in their products."

Commit to Quality, Reduce Risk
Well-publicized software failures in recent times have been spectacular. We want these failures to become the exception instead of the norm. We want to encourage a thriving industry that easily enables quality work.

“Well-publicized software failures in recent times have been spectacular. We want these failures to become the exception instead of the norm. We want to encourage a thriving industry that easily enables quality work to happen.”
The Application Security Industry Is Now Bigger Than The Application Development Industry
Common Misconceptions -1

We must start with firm requirements
If it passes test, it must be OK
Software quality can’t be measured
The problems are technical
We need better people
Software management is different

_Managing the Software Process_, Watts Humphrey, Addison
Common Misconceptions – 2

Maturity levels guarantee results
Maturity level 3 is all that is needed
Higher maturity levels add to cost
Higher maturity levels are needed only for safety critical or business mission critical systems
If it is “agile” or “lean”, it is good
What we need are lean processes
Maturity level 5 is the end
The Real Question

Whose Process Is It?
Why? - 1

Why do development teams agree to **delivery schedule** they know they can’t meet?

Why don’t C-level executives realize that poor **quality performance is the root cause** of most software cost and schedule problems?

Why doesn’t the government **hold contractors liable** for software defects and vulnerabilities?
Why does the software applications development industry believe that quality increases costs and schedule?

Why do we continue to rely on test as the principal defect removal method?

Why do we continue to rely on monthly status reporting when projects get to be one year late one day at a time?
Why? - 3

Why don’t we call technical debt for what it really is, “malpractice”?

Why don’t we charge the cost of post release bug fixing (corrective maintenance) to development where it belongs?

Why do we approach software and supply chain assurance as a technical problem and not the management problem that it is?
If the next Pearl Harbor is going to be a cyber attack

Should we call software bugs, software bombs?
Have You Considered?

Quality work is more **predictable**

**Unhappy** people rarely do quality work

Without quality, agility is in **name only**

Quality **without numbers** is just talk
Immutable Laws of Software Development – 1

The number of development hours will be directly proportional to the size of the software product.
Immutable Laws of Software Development – 2

When acquirers and vendors both “guess” as to how long a project should take, the acquirers’ “guess” will always win

• Customers’ Dilemma
  • Want their product now at zero cost.
  • Due to time-to-market pressures, time frames are arbitrary and unrealistic for the software team to produce a product that works.

• Developers’ Choices
  • Try to “guess” what it would take to win the business.
  • Or make a commitment based on a plan and what the organization can do based on organization historic data.
Immutable Laws of Software Development – 3

When management compresses schedule arbitrarily, the project will end up taking longer

<table>
<thead>
<tr>
<th></th>
<th>Default</th>
<th>10% Compression</th>
<th>20% Compression</th>
<th>10% Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration Mths</td>
<td>25.9</td>
<td>23.3</td>
<td>20.7</td>
<td>28.5</td>
</tr>
<tr>
<td>Defect Count</td>
<td>1,033</td>
<td>1,316</td>
<td>1,715</td>
<td>849</td>
</tr>
<tr>
<td>% Change</td>
<td></td>
<td>27.4%</td>
<td>66.0%</td>
<td>-17.8%</td>
</tr>
</tbody>
</table>
When poor quality impacts schedule, schedule problems will end up as quality disasters

Maryland officials were warned for a year of problems with online health-insurance site

"We didn't know it would be broken when we turned it on"
Immutable Laws of Software Development – 6

The less you know about a project during development, the more you will be forced to know later

<table>
<thead>
<tr>
<th>Data for week of</th>
<th>26-Mar-12</th>
<th>24 of 52</th>
<th>PROJECTED END DATE</th>
<th>Week Of</th>
<th>Week(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Avg EV Eff/Wk</td>
<td>6-Aug-12</td>
</tr>
<tr>
<td>Baseline Plan</td>
<td>479.0</td>
<td>485.4</td>
<td>Actual Plan</td>
<td>1.01</td>
<td></td>
</tr>
<tr>
<td>Project Hours</td>
<td>9910.0</td>
<td>10253.4</td>
<td>1.03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Project Hours To-Date</td>
<td>2.20%</td>
<td>2.60%</td>
<td>1.18</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Earned Value</td>
<td></td>
<td></td>
<td></td>
<td>Top 8 Avg EV Eff/Wk</td>
<td>30-Jul-12</td>
</tr>
<tr>
<td>EV To-Date</td>
<td>51.80%</td>
<td>50.20%</td>
<td>0.97</td>
<td></td>
<td></td>
</tr>
<tr>
<td>EV Effort %</td>
<td></td>
<td></td>
<td></td>
<td>Blocked EV Effort</td>
<td>732.3</td>
</tr>
<tr>
<td>Cost of Quality [(A+FR+PREV)/TOTAL EFFORT]</td>
<td>3317.9</td>
<td>32.4%</td>
<td></td>
<td>Avg EV / Week</td>
<td>1.8</td>
</tr>
<tr>
<td>Engineering Effort To-Date</td>
<td>8379.8</td>
<td>81.7%</td>
<td></td>
<td>Avg EV Effort / Week</td>
<td>244.8</td>
</tr>
<tr>
<td>Management Effort To-Date</td>
<td>1873.6</td>
<td>18.3%</td>
<td></td>
<td>Total EV Effort Required</td>
<td>6,855.6</td>
</tr>
</tbody>
</table>
Immutable Laws of Software Development – 7

When test is the principal defect removal method during development, corrective maintenance will account for the majority of the maintenance spend
Immutable Laws of Software Development – 8

The number of defects found in production use will be inversely proportional to the percent of defects removed prior to integration, system, and acceptance testing.
Immutable Laws of Software Development – 9

The amount of technical debt is inversely proportional to the length of the agile sprint

<table>
<thead>
<tr>
<th>Measure</th>
<th>S1</th>
<th>S2</th>
<th>S3</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>System test defects</td>
<td>21</td>
<td>25</td>
<td>34</td>
<td>System test defects includes all defects found post unit test</td>
</tr>
<tr>
<td>High severity system test defects</td>
<td>16</td>
<td>6</td>
<td>8</td>
<td></td>
</tr>
<tr>
<td>Open high severity system test defects</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Defects not closed at Sprint end</td>
</tr>
<tr>
<td>Open low severity system test defects</td>
<td>3</td>
<td>3</td>
<td>14</td>
<td></td>
</tr>
<tr>
<td>Peer review defects</td>
<td>4</td>
<td>3</td>
<td>7</td>
<td>Major operational defects only</td>
</tr>
<tr>
<td>System test defect density – high severity defects</td>
<td>1.14</td>
<td>0.84</td>
<td>0.66</td>
<td></td>
</tr>
<tr>
<td>System test defect density – total defects</td>
<td>1.5</td>
<td>3.5</td>
<td>2.8</td>
<td></td>
</tr>
<tr>
<td>%Early defect removal</td>
<td>16%</td>
<td>11%</td>
<td>17%</td>
<td>Defects found in Peer Reviews/Total Defects found</td>
</tr>
<tr>
<td>Net Code Churn (LOC)</td>
<td>13979</td>
<td>7115</td>
<td>12154</td>
<td>Measured by taking snapshots of code at beginning and end of Sprint, and then diffing the snapshots</td>
</tr>
</tbody>
</table>
Immutable Laws of Software Development – 12

Insanity is doing the same thing over and over and firing the project manager or the contractor when you don’t get the results you expected
Results
Organization History

Constancy of Purpose
• Make schedule and quality predictable. Since the introduction of HVD, average schedule deviation has been less than 5%

Focus on quality:
• Removal of defects at the earliest opportunity, before test where they are the least costly to remove
• Quality is more predictable
• Unhappy people rarely do quality work

On the project for the Selective Service System, we were able to deliver 680,000 lines of source code where:
• Zero security vulnerabilities were found in pen testing
• Production deployment 2 weeks ahead of schedule
• Schedule deviation less than 2% throughout 150 weeks of development
• Zero system downtime in over 3 years of production use due to software defects
Results
Recently Completed Project

Component yield: 92.3%
  • Percent of defects introduced during development that were removed during development (before integration or system test)

Cost of Quality: 34.9% [Industry average: >50%]
  • Effort in Appraisal, Failure and Prevention tasks

Time to Accept Deliverables:
  • 1.3 Weeks per 100,000 SLOC [Industry average: >16 Weeks]
  • 0.21 Defects/KLOC [Industry average: 4.73]

Schedule deviation: 4 weeks ahead of schedule
  • 2.5% ahead [Industry average: 27% behind]
Results
New Team Member

43 Components
Size estimate error: 9%
Effort estimate error: 13%
Process Quality Index (PQI): 0.73
  SEI data: PQI > 0.4 indicates high quality component
Component yield: 93.5%
  Percent of defects introduced during development that were removed during development (before integration or system test)
# Performance Metrics That Matter

## Benchmarking

<table>
<thead>
<tr>
<th>Metric</th>
<th>Industry Average</th>
<th>Company Average</th>
</tr>
</thead>
<tbody>
<tr>
<td>Schedule deviation</td>
<td>&gt; 50%</td>
<td>&lt; 6%</td>
</tr>
<tr>
<td>No. of defects in delivered product</td>
<td>&gt; 100</td>
<td>&lt; 15</td>
</tr>
<tr>
<td>(Size: 100,000 Source Lines of Code)</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Customer’s time to accept 100,000 SLOC</td>
<td>&gt; 4 Months</td>
<td>&lt; 5 Weeks</td>
</tr>
<tr>
<td>product</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% of design and code inspected</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>% of defects removed prior to system test</td>
<td>&lt; 60%</td>
<td>&gt; 85%</td>
</tr>
<tr>
<td>% of development time fixing system test</td>
<td>&gt; 33%</td>
<td>&lt; 10%</td>
</tr>
<tr>
<td>defects</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Cost of quality</td>
<td>&gt; 50%</td>
<td>&lt; 35%</td>
</tr>
<tr>
<td>Warranty on products</td>
<td>?</td>
<td>Lifetime</td>
</tr>
</tbody>
</table>
Agile Project Management Example

Senior Management View

Agile Project Team View

Release Burndown

Issues / Risks

Daily Burndown

Release & Sprint Velocity

Sprint Taskboard
## “Vital Few” Performance Metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Increment</th>
<th>Sprint</th>
<th>Component</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planned vs. Actual size, effort, schedule, earned value</td>
<td>√</td>
<td>√</td>
<td>√</td>
</tr>
<tr>
<td>Cost of quality - % development effort in defect prevention, pre-test defect removal, testing defect removal, post-release defect removal</td>
<td>√</td>
<td></td>
<td></td>
</tr>
<tr>
<td>% defects removed prior to system test</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Time in User Acceptance Test</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>% with zero post-unit test defects</td>
<td></td>
<td></td>
<td>√</td>
</tr>
<tr>
<td>% design, code inspected</td>
<td>√</td>
<td>√</td>
<td></td>
</tr>
<tr>
<td>Process improvement proposals</td>
<td>√</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Government
Expect More

Make **quality** the number one goal

Hold contractors **liable** for software defects or vulnerabilities

Acquire **Lowest Price Guaranteed Quality** (LPGQ) offers rather than Lowest Price Technically Acceptable (LPTA) or Best Value offers

**Trust** contractors but verify
Industry
Be Responsible for Quality

Make **quality** the number one goal

Cease dependence on test and rework for **defect removal**

Provide **quality guarantees** while continually improving cost and schedule performance

Support **2013 NDAA Sec 933**
Empower Developers

End the practice of imposing *arbitrary* and unrealistic schedules

**Trust** and support the teams

Train software developers to negotiate *realistic and aggressive* schedule

Have Fun on the Job
Joy in Work

“There is a square; there is an oblong. The players take the square and place it upon the oblong. They place it very accurately; they make a perfect dwelling place. Very little is left outside. The structure is now visible; what was inchoate is here stated; we are not so various or so mean; we have made oblongs and stood them upon squares. This is our triumph; this is our consolation.”

The players in Virginia Woolf’s *The Waves*
What does “FUN ON THE JOB” Mean to you?
“If I have made myself too clear, you must have misunderstood me”
Alan Greenspan

Questions?
Contact

Girish Seshagiri

girish.seshagiri@ishpi.net

703 426-2790
Performance Metrics That Matter: Eliminating Surprises in Agile Projects

Girish Seshagiri, Ishpi Information Technologies, Inc.

Software Solutions Conference 2015
November 16–18, 2015