Security Changes Everything

Watts S. Humphrey
Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Sponsored by the U.S. Department of Defense
© 2004 by Carnegie Mellon University
Agenda

The security problem

The central nature of quality

Disciplined development methods

Quality results

Quality and plan management

Quality work pays
The Security Problem

In the old days, we could assume that our users were
• friendly
• interested in results
• willing to help

The Internet used to be like the old days, but no more.

The security situation is bad and getting worse.
• There are lots of bad folks, and more each year.
• They are getting smarter all the time.
• They haven’t turned vicious yet, but they will.
Security Incidents

Total incidents reported (1988-2003): **319,992**. An incident may involve one or thousands of sites and incidents may last for long periods.

Source: CERT/CC
The Response Strategy Is Failing

The response strategy accepts the cost of an initial attack.

It is impractical for system administrators.

It is expensive for suppliers.
  • excessive development and repair costs
  • unknown and possibly unlimited litigation exposures

It is troublesome for users.

The response strategy cannot consistently or economically produce secure software.
The Central Nature of Quality

Many lives and businesses now depend on software.

We need larger, more complex, more secure, and safer software systems on predictable schedules.

The need for highly reliable, safe, and secure software will increase.

Poor-quality software cannot be safe, secure, or reliable.

We have reached the limits of traditional software practices.
Delivered Defects by CMM Level

Defect Density of Delivered Software

<table>
<thead>
<tr>
<th>CMM Level</th>
<th>Defects/KLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>7.5</td>
</tr>
<tr>
<td>2</td>
<td>6.24</td>
</tr>
<tr>
<td>3</td>
<td>4.73</td>
</tr>
<tr>
<td>4</td>
<td>2.28</td>
</tr>
<tr>
<td>5</td>
<td>1.05</td>
</tr>
</tbody>
</table>

Ref: SEI Technical Report 2003-014
Current Defect Levels

While 1.05 defects per KLOC may seem good, that is 1,050 defects per million lines of code (MLOC).

With the multi-million LOC systems in general use today, such defect levels are unacceptable.

To produce safe and secure systems, we need defect levels of 1 to 10 defects per MLOC.

The current test-based software quality practices cannot be improved by 100 to 1,000 times.
The Limitations of Testing

Large complex programs cannot be exhaustively tested.
• It is impossible to test every operating condition.
• Testing must focus on the most frequent conditions.
• Extensive user testing then finds more defects.

Testing finds less than 50% of the defects in most products.

The fundamental problem is that system-level testing tests the paths through the system.

To get most of the defects, you must test most of the possible paths.
Possible Test Paths

A 4 by 4 Maze

Possible Maze Paths

<table>
<thead>
<tr>
<th>Size</th>
<th>Switches</th>
<th>Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>3</td>
<td>9</td>
<td>20</td>
</tr>
<tr>
<td>4</td>
<td>16</td>
<td>70</td>
</tr>
<tr>
<td>5</td>
<td>25</td>
<td>252</td>
</tr>
<tr>
<td>10</td>
<td>100</td>
<td>184756</td>
</tr>
<tr>
<td>20</td>
<td>400</td>
<td>1.38E+11</td>
</tr>
</tbody>
</table>

Any untested path could have defects.
Why do Defective Systems Work?

- Overload
- Hardware failure
- Configuration
- Resource contention
- Operator error
- Data error

Secure region = tested (shaded)
Insecure region = untested (unshaded)
Software Inspections

Since test-based methods cannot find most defects, advanced software groups use inspections.

Inspections are effective but inherently limited.
• A 1 KLOC program listing has 30 pages.
• 1 defect/KLOC is 1 defect in 30 pages.
• 100 defects per MLOC is 1 defect in 300 pages.
• 10 defects per MLOC is one defect in 3,000 pages.
• 1 defect per MLOC is one defect in 30,000 pages.

One defect in 30 pages is already near the limit of human capability.
Next Steps

To achieve improvements of 10, 100, or 1,000 times, it is instructive to look at other technologies.

No other technology uses test-based quality practices. They use
• disciplined development processes
• extensive quality data
• data gathering by the workers
• statistical quality control

To make significant improvements, our developers must have the personal discipline to work this same way.
Disciplined Development Methods

Several methods already focus on the developers’ personal disciplines.

Clean Room – developed in the 1970s by IBM
• each developer uses formal design methods
• quality is managed before test
• quality results are ten times better than level-5 groups

Correct-by-Construction (CBC) – Praxis, England
• rigorous quality management by team members
• well-defined and tightly managed development process
• quality levels at least ten times current level-5 results

The TSP, developed by the SEI, has achieved comparable results.
The Required Practices

Although they are a necessary first step, we need more than just disciplined software developers.

We need disciplined development teams.

The team members must all use disciplined personal practices and the teams must be
• highly motivated
• committed to quality work
• have management support

This requires rethinking our engineering management system.
Large-scale Creative Work

Software development is intellectual work and its challenges are similar to but different from other technologies.

In other fields, the quality of each individual’s work is often obvious.

In software, the developers’ personal practices are a private concern.

If they want to follow the organization’s defined practices, they will, but nobody will know if they don’t.
Large-scale Work Paradigms

The paradigms for large-scale work are armies and factories.
• individual performance is usually visible
• supervision is often autocratic

This will not work for software: to do quality work, the developers must care about the quality of their work.

This requires a quality attitude and work ethic where
• the quality of every member’s work is a team concern
• every team member gathers and uses quality data
• the team uses these data to control product quality
Self-directed Teams

To do consistently disciplined work, developers must be on self-directed teams.

Such teams are motivated to produce quality products on schedule.

Self-directed teams
• establish their own goals and practices
• produce their own plans
• negotiate their own commitments
• track and report on their own work
• manage the quality of their own products
• consistently meet their own commitments
Management Support -1

The need is to convince management that their developers should work on self-directed teams.

Management will then support the developers as long as they
• strive to meet management’s needs
• convince management that their plans are sound
• do quality work
• regularly report on their work
• respond to changing needs
• come to management for help when they have problems
Management Support -2

This is a bargain.

Management will agree to self-directed teamwork as long as they believe the teams are doing a superior job.

To do that, the developers must
• maintain precise and accurate plans
• measure and track their work
• control product quality
• regularly show management that they are doing superior work

The TSP shows developers how to do this.
Quality Results

Many TSP teams have been launched and the results are impressive.

TSP teams range from single teams in one location to multiple teams in several cities or countries.

One successful TSP team is in Redmond, Washington and Hyderabad, India.

Another has groups in two different cities with members from competing companies.
TSP Quality Levels

Defect Density of Delivered Software

- CMM Level 1: 7.5
- CMM Level 2: 6.24
- CMM Level 3: 4.73
- CMM Level 4: 2.28
- CMM Level 5: 1.05
- TSP: 0.06

Ref: SEI Technical Report 2003-014
Quality Work Pays

Today’s software professionals are capable of extraordinary work.

Once they know and use the proper methods, they routinely produce products with practically no defects.

What is truly exciting is that these methods also save time and money.

Of 24 TSP teams studied to date, the average productivity improvement was 78%.
Developer Reactions

With proper leadership and support, TSP teams can produce extraordinary results.

Developers like to use the TSP:

- This really feels like a tight team.
- The TSP forces you to design, to think the whole thing out.
- Design time is way up but code time decreased to compensate.
- Tracking your time is an eye opener.
- Really good teamwork on this project - no duplication of effort.
- I’m more productive.
- Gives you incredible insight into project performance.
- Wonderful to have team members assigned specific roles.
- Team really came together to make the plan.
- I feel included and empowered.
TSP Users

Some of the organizations using the TSP are

- ABB
- Bechtel
- Boeing
- Census Bureau
- DFAS
- EDS-SDRC
- Erickson
- Honeywell
- IBM Japan
- Intuit
- Lockheed
- Microsoft
- NASA Langley
- Northrop Grumman
- Teradata
- USA: AMCOM
- USAF: Hill AFB
- USN: NAVAIR
- USN: NAVOCEANO
- Vicarious Visions
- Xerox
Conclusion

Security is a quality problem.

To fix the security problem, the top development priority must be quality.

To consistently build quality software, we must change software-development practices.

This requires self-directed development teams and management support.
For More Information

Visit the PSP or TSP web sites
http://www.sei.cmu.edu/psp/
http://www.sei.cmu.edu/tsp/

Contact a PSP transition partner
http://www.sei.cmu.edu/collaborating/partners/trans.part.psp.html

Contact SEI customer relations
Software Engineering Institute, Carnegie Mellon University
Pittsburgh, PA 15213-3890
Phone, voice mail, and on-demand FAX: 412/268-5800
E-mail: customer-relations@sei.cmu.edu

See the book
Winning With Software: an Executive Strategy, by Watts Humphrey, Addison-Wesley, 2002