6-17-2016

Metrics, Software Engineering, Small Systems – the Future of Systems Development

William L. Honig

Loyola University Chicago, whonig@luc.edu

Recommended Citation
Honig, William L.. Metrics, Software Engineering, Small Systems – the Future of Systems Development. , , , 2016. Retrieved from Loyola eCommons, Computer Science: Faculty Publications and Other Works,
What I Hope You Will Remember:
1. Trend to smaller and smaller computing devices will continue
2. Quality, Reliability, Trustworthiness of computer systems will increase in importance
3. Metrics, and good software engineering are key to it all

Outline
1. Where I Started = Early Metrics
2. Metrics Today
3. What are "Embedded Systems" – Where / What Today
5. Why Good Software Engineering is Essential
6. Summary Thoughts

What I want YOU to do!
1. Questions are GOOD!
2. You can ask anytime.
3. It is not BAD to ask questions or make comments
   - It does not mean "I don’t understand"
   - It does not mean "I am stupid"
4. It is GOOD to ask questions
   - Shows you are awake
   - Shows you are interested
   - Help others understand too!

I am expecting you to ask questions ANY TIME!!
whonig@luc.edu

My First Computer
– Instruction Speed: 1 instruction every 2 msec.
– QUIZ: What speed processor would this be in today’s terms (Ghertz)?

Quiz 1

IBM 7094
1 instruction every .002 seconds
1/.002 = 500 instructions a second
(assume 1 clock cycle for instruction)
Today Micycles or Gcycles per second
500/10^14

Programming Tools
– One punched card per line of program (72 characters)
My First Job – Bell Labs

- Computer controlled telephone switching
- Reliability and performance
- EVERYTHING is measured = METRICS
- Metrics can be used for GOOD and BAD

Metrics:
1. Seconds to complete a call
2. Number of customers an hour
3. Customers who call back quickly
4. ...

I did not know at this point how important it would be!
whonig@luc.edu

Outline
1. Where I Started = Early Metrics
2. Metrics Today
5. Why Good Software Engineering is Essential
6. Summary Thoughts

What are Metrics?

Measures, Quantitative Values, Numbers

Things you need or want to measure

IEEE Standard Glossary of Software Engineering Terminology
Std 610.12 -1990:
Metric. A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Food Metrics

Things you may want to know before buying a food item

Example Metrics:
1. Protein per servings
2. Salt (NaCl) per serving

What are some typical software development metrics?

SLOC, KSLOC - Source Lines of Code (may distinguish new, reused, changed, ...)

Person Hours - Actual Time Worked (e.g. on coding, or on whole development project)

Defects - Count of Bugs or Problems Found (tracking where defects are found and where they were caused is key to process improvement)

Earned Value - A Measure of Performance to Schedule

AFR - Appraisal to Failure Ratio (comparing time spent preventing bugs to time spent fixing them)

What are Metrics?

Measures, Quantitative Values, Numbers

Things you need or want to measure

IEEE Standard Glossary of Software Engineering Terminology
Std 610.12 -1990:
Metric. A quantitative measure of the degree to which a system, component, or process possesses a given attribute.

Food Metrics

Things you may want to know before buying a food item

Example Metrics:
1. Protein per servings
2. Salt (NaCl) per serving

What are some typical software development metrics?

SLOC, KSLOC - Source Lines of Code (may distinguish new, reused, changed, ...)

Person Hours - Actual Time Worked (e.g. on coding, or on whole development project)

Defects - Count of Bugs or Problems Found (tracking where defects are found and where they were caused is key to process improvement)

Earned Value - A Measure of Performance to Schedule

AFR - Appraisal to Failure Ratio (comparing time spent preventing bugs to time spent fixing them)

Food Metrics

Things you may want to know before buying a food item

Example Metrics:
1. Protein per servings
2. Salt (NaCl) per serving

History of System Development

1960 → 1990

Software Art
Software Science
Software Process
Why bother?
If you don’t know where you’re going, any road will do!

Questions:
- How good is my system (right now)?
- How good is my software process (right now)?
- What must I do to improve it?
- Where do I start?
- What am I doing well (and not so well)?
- Am I on schedule?
- Is the system high quality?
- Will it be high quality when finished?

The general quality process:
- Measure (something(s))
- Set Targets, Goals
- Try to Improve to Meet Goals
- Do it again (and again...)

The quality improvement process PDCA

In the USA:
- Much work on product quality is motivated by the Japanese success in high technology manufacturing.
  - Automobiles
  - Cameras
  - ...

The Quality Process and Metrics
(two parts of a whole)

Defined Process; Repeatable Process; Quality Process
- Known steps, known inputs and outputs, entry and exit criteria
- Cost to remove defects or correct mistakes doubles each step further into the development
- Feedback: defect reporting, cause analysis, corrective action plan

Measurement and Metrics
- Data gathering for understanding, evaluation, control, prediction. (Data gathering can be expensive)
- Metrics can be objective or subjective

SSTSPI Product Summary
Team (KingMe), Cycle (3), 4/13/2015

Want to know more?
As a real world software engineer, you should...
- See more on SEI
  http://www.sei.cmu.edu/about/index.cfm
- Learn more on quality process
  https://asq.org/learn-about-quality/total-quality-management/overview/deming-points.html
I soon realized that this kind of computer work was different!

Outline
1. Where I Started = Early Metrics
2. Metrics Today
5. Why Good Software Engineering is Essential
6. Summary Thoughts

My First Programming Work (Bell Labs)

My First Keyboard

My First Programming Work (Bell Labs)

Metrics == Run Time (for reliability and quality)

- Lights out system, watches itself
  - Part of the software does the "job"
  - Other parts watch for hardware failures
  - Other parts correct software failures
  - Measure (Count and Keep)
- Part of the software does the "job"
- Measure (Count and Keep)
- Other parts watch for hardware failures
- Other parts correct software failures
- Measure (Count and Keep)
- Health of whole system more important than most individual tasks
  - Ignore a request for a new phone call, but keep the whole system running

Quiz: How long can a 5 9's system be down in a week (or in a year)?

Quiz 2

Availability
This includes:
- Hardware problems
- System upgrades
- Power Failures
- Bugs

Also
Never write code without knowing how long it might take to run

What a wonderful world it will be....
Quiz - What Know About Small Systems and Mobility?

Who / When First Tablet Computer?

Who / When First Smart Phone?

Early Adopters / Missionaries / Pioneers...

Archos (French)

Palm Treo

Windows 7, Touch On Screen Keyboard Stylus

PalmOS, Touch Graffiti and Cursive Stylus, and PHONE

2006

2003

Small Systems - Wearables

- Personal Area Network or Wearable Area Network
- Things we keep around us and use to do what we do
- Likely to become much smaller than a phone or tablet
- Increased Importance
  - Metrics
  - Quality
  - Reliability

Power? “Hey, can you spare a charge?”

Small Systems

Shrinking Computer parts means small systems

Examples

Toaster

Home appliances

https://juneoven.com/

These systems change the kind of software developers need to know

Broad Framework

Architecture

Components and Communications

Soft to Hard Real Time

Statistical Network Performance

Virtual and Physical Service

Full Networks, Internet Application Architectures

Dynamic App Evolution

Enhanced Security

Networks will change

Internet of Things will NOT use “just” today’s internet

Security, capacity, reliability, power needs...

Outline

1. Where I Started = Early Metrics
2. Metrics Today
5. Why Good Software Engineering is Essential
6. Summary Thoughts

What is needed to make it all work?
Early Anti-Computer Movement

Early reaction to automated ("computer driven") manufacturing ("embedded systems")

Ned Ludd, 1812

Luddites Destroying Loom

Luddites Today??

Do Computers Help or Hurt?

Code sheets were calculated by hand by women "computers" Is it bad they lost their jobs? Replaced by machines....

Alan Turing

Bletchley Park

Luddites Today??

Does Software Engineering Work?

Dr. William L Honig

Many definitions exist, but the core is:

"The study of the development (including maintenance) of software of high quality in a highly productive manner."

「質のよいソフトウェアの効率よい開発、およびその運用・保守を扱う学問」

Definition

Software Process

Policies, techniques, procedures, etc for developing software
Activities such as analysis and design
Software is normally developed by a team.
Not by just one individual.
Need to manage the team.
→ Need to define the process.
→ Process may need to be defined per organization.

Why Software Engineering

Problems:
● Systems Late
● Incomplete
● Buggy
● No one knows when it’s "done"

If you don’t know where you are going, any road will get you there.
Paraphrase of exchange between whom?
Hint: Lewis Carroll

Possible Solutions:
➤ Requirements
➤ Analysis & Design
➤ Metrics and Measures
➤ Continuous Quality Improvement

The disciplined development of great computer based systems for the world!
Quiz 4

Reading ➔
(1865.) Alice’s Adventures in Wonderland; Through the Looking Glass
(1995) Neal Stephenson, The Diamond Age: Or, A Young Lady's Illustrated Primer

Math and programming fundamentals
Nano technology, virtual reality
Post oil, biotechnology

A Growing Problem

• Software that has hidden features
  – spyware
  – Unexpected functions and impacts
• Why?
  – Malicious intend (a whole other issue)
  – Poor systems thinking and analysis

Software Transparency and Purity

Transparency: All functions are disclosed to the users / owners / operators of the system
Purity: system does nothing irrelevant to its stated purpose, nothing foreign to it’s advertised nature

For more details see Pascal Meunier, Software Transparency and Purity, Communications of the ACM (51,2) Feb 2008.

Quiz 5

A team of programmers has been working hard to finish a system. They have written 2347 lines of Java code over the last 3 months.

They have been testing for the last two weeks.
So far they have found and fixed 23 bugs

How many more bugs may be in the system?
Are they finished testing? Or should they keep working? Is the system finished and ready to release?

Management ➔
What if the future of the company depends on this system coming out on time and with good quality?

Defects ➔
Another word for bugs, errors, mistakes.

Answer ➔
It’s impossible to tell how many bugs remain

BUT!
Good software engineering + quality processes can solve it

Metrics that can give answers:
  Defect Density (past and similar projects)
  Defect Arrival Rate and Defect Fix Rate
  Cost of Rework (Defects caused by other fixes)
  Capture / Recapture Calculation (Inspections)

YOU ➔
Don’t you want to be able to do this???

Believe Me...

Maturity to use Metrics and Software Engineering Process
• Alternative is chaos, heroes, burnout, no predictability

Democratic Development Teams
• Teams can control their own destiny, schedule, results, rewards...
• No need to guess (schedules, results, quality)
Outline
1. Where I Started = Early Metrics
2. Metrics Today
5. Why Good Software Engineering is Essential
6. Summary Thoughts

What does this mean to computer science students today?

Learn the difference between Great, Good, and OK programming
• Even more important for small systems
• To me, this requires metrics, good software engineering process

Growth opportunities in
• Reliable, secure, trustworthy systems
• Small systems, their networks and security

Three waves of computing systems….

Big Computing
• Large dedicated mainframes (eventually mini computers)
• Large and reliable embedded systems
• Global access to applications
• Large programming organizations

Personal Computing
• Go to computer to do work
• Democratic applications
• Quality and Reliability
• Suffers; Defects Acceptable
• More, smaller, quicker programming teams

Pervasive Computing
• Devices with us all the time (in us?)
• Devices work “on their own”, talk to each other
• Who and how will the software be made?

This is YOUR future. Be Ready for It.