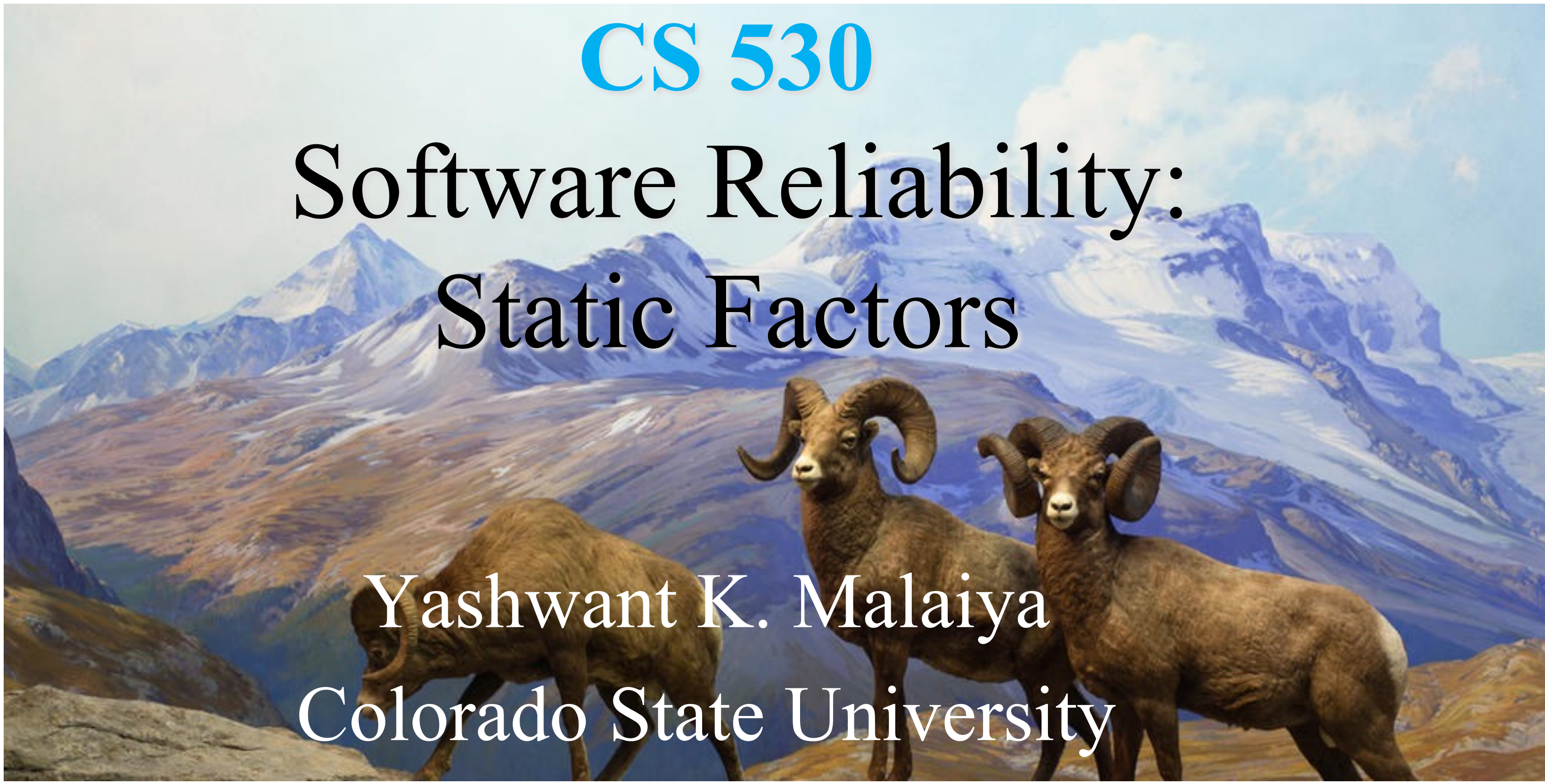
Fault Tolerant Computing



CS 530

Software Reliability: Static Factors

Yashwant K. Malaiya
Colorado State University

Outline

- Techniques available in Software Reliability
- Software & Hardware Reliability
- Defect density & factors that control it
  - Phase
  - Programming team and process maturity
  - Software Structure
  - Requirement volatility
  - Reuse

Cost of Poor Software Quality

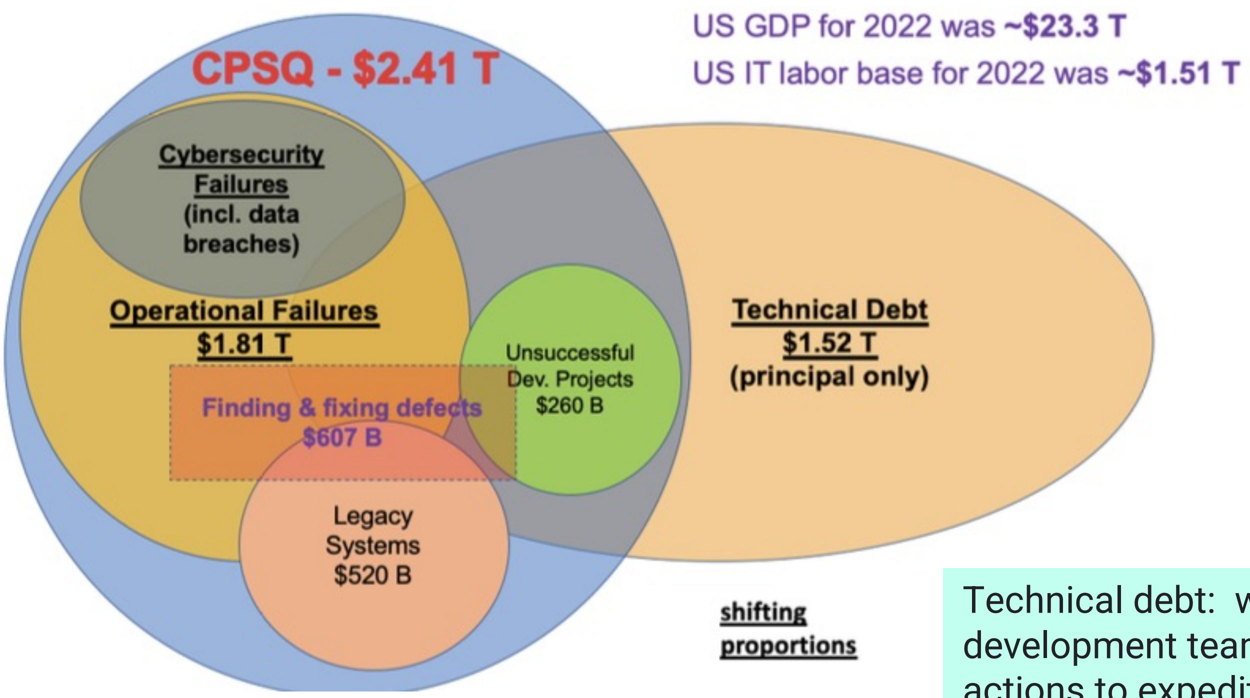
For the year 2022, the total Cost of Poor Software Quality (CPSQ) in the US is $2.41 trillion (T).

- The largest contributor to CPSQ is operational software failures., $1.81 T.
- Cost of finding and fixing defects $607 B.
- Unsuccessful development projects $260 B.
- Legacy system problems contributed $520 B.

The Cost of Poor Software Quality in the US: A 2022 Report

Elon Musk Becomes First Person Ever to Lose $200 B.
12/22
Adani had lost 60 B.

Cost of Poor Software Quality

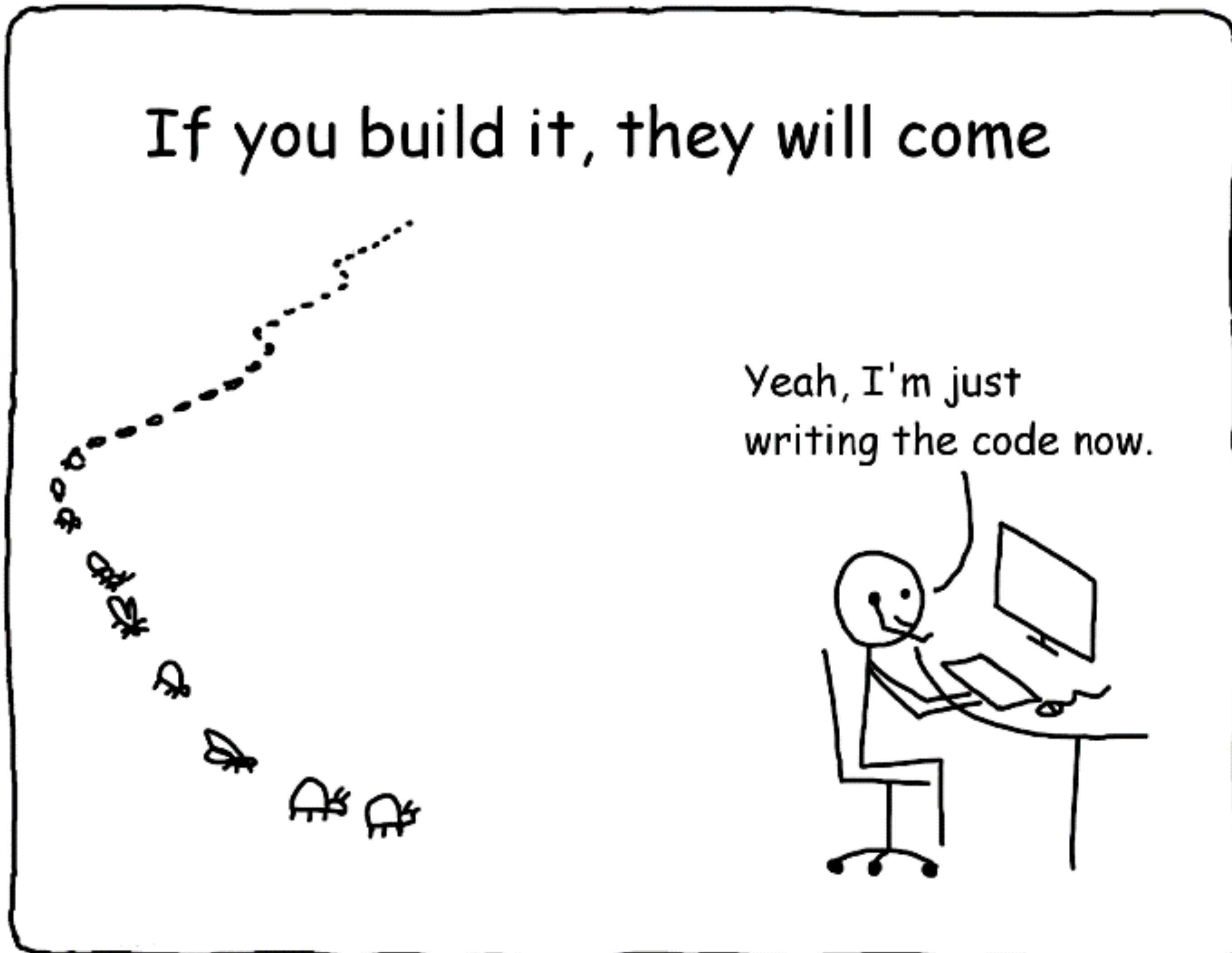


The Cost of Poor Software Quality in the US: A 2022 Report

Technical debt: when development teams take actions to expedite the delivery of code which later needs to be refactored.

Time to go Wholistic

- We have data on different aspects of reliability to have reasonable hypotheses.
- We know limitations of the hypotheses.
- We have enough techniques & tools to start engineering.
- Accuracy **comparable to or better** than established hardware reliability methods.



cartoontester.blogspot.com © 2013

History's Worst Software Bugs

Why It's Needed Now

- Reliability expectations growing fast
- Larger projects, shorter time
- Quick changes in developing environments
- Reliance on a single technique not enough
- Pioneering work has already been done.

Why It's Time: Emergence of SRE

- **Craft**: incremental intuitive refinement
- **Science**: ***why*** it is so
  - Observe, hypothesize, assess accuracy
- **Engineering**: ***how*** to get what we want
  - Approximate, integrate, evaluate
- Are we ready to engineer software reliability?

Learning from Hardware Reliability

- Hardware Reliability Methods: Well-known, well-established methods
  - Now standard practice
  - Used by government and industrial organizations worldwide
  - Considered a *well-established science*

Hardware Reliability: The Status (1)

- Earliest tube computers: MTTF comparable to some computation times!
- 1956 RCA TR-1100: component failure rate models
- 1959: MIL-HDBK-217A: common failure rate: 0.4×10^{-6} *for all ICs for all cases*
  - Revised about every 7 years until 95. Addition ANSI/VITA-51.1 - 2013 R18.

Hardware Reliability: The Status (2)

- Why use hardware reliability prediction?
  - Feasibility Study: initial design
  - Compare Design Alternatives: Reliability along with performance and cost
  - Find Likely Problem Spots- high contributors to the product failure rate
  - Track Reliability Improvements

Hardware vs Software Faults

- Hardware faults are generally field or manufacturing process defects.
- Software faults are due to incorrect design/implementation (“man-made”).
- During debugging, bugs are removed thus reliability grows.
- Design defects on hardware are basically similar to software defects.

Hardware vs Software Reliability Methods: Use of models

	Model selection based on	Parameters estimated using
Hardware	Past experience with similar units	Past experience with similar units
Software	Past experience* with similar units	**Early:** past experience with similar units **Later:** from the unit under test

* Some researchers have suggested model selected using early test data from the software under test.

Bugs May Delay Dbase IV Shipment to Fall, Ashton-Tate Says

By Mark Brownstein and Scott Mace

LOS ANGELES — Ashton-Tate told shareholders last week that the still bug-ridden Dbase IV could miss its projected July shipping date but will definitely be out by September 30.

Blaming the possible delay on undiscovered bugs, Ashton-Tate chairman and CEO Edward J. Esber Jr. said, "We can fix all the bugs that we've already found before July," and he suggested Ashton-Tate could ship Dbase IV in July "if we don't find any more new bugs."

"Finding and fixing bugs is the only issue" that could cause further delay, said Roy E. Folk, Ashton-Tate vice president and general manager of software development, at the annual shareholders' meeting. Folk challenged speculation that the product takes most of the 640K of available RAM under DOS, saying it can run a 100K program on a network.

In fact, Ashton-Tate is fixing bugs by correcting code errors instead of using program patches that add to the program's size, Folk said.

Esber also spoke of other upcoming Dbase IV versions, including a Presentation Manager release and Dbase IV, Version 1.1, which will take advantage of the Ashton-Tate/Microsoft SQL Server announced in January.

"Dbase IV/PM will feature a graphical user interface," Esber said. "You will also see further extensions in language and some additional functionality."

"The direction we're taking beyond Dbase IV should maintain our lead in the Dbase area," Esber said.

While chief financial officer George L. Farinsky said Ashton-Tate could weather any sales losses resulting from the later shipping date, developers of Dbase add-ons are faring considerably poorer. Sources close to Wallsoft Systems of New York say the developer of the Dbase code generator and template language UI Programmer is for sale. The firm has been hurt by the announcement that Dbase IV will include a UI-like code generator and template language. Although Wallsoft plans to release a second-generation update superior to that announced by Ashton-Tate, it still awaits Dbase IV.

"We've got 500 of the Fortune 500 companies saying, 'Your product looks good, but we want to wait to see Dbase IV,'" said Martin Rinehart, Wallsoft's president. Rinehart, who earlier this year led an independent developers' effort to create a Dbase standard, would not comment on reports of his company's troubles. He was among Dbase add-on developers who joined Ashton-Tate at the Dbase IV announcement and are now being hit hard by Dbase IV's delay.

Bytel Corp. of Berkeley, California, also acknowledged orders are "down considerably" for its Dbase code generator, Genifer.

None of the add-on vendors greeted last week's announcement of a further Dbase IV delay, though some claim business is still good. "It's been for us the best spring ever," said John Henderson, president of Concentric Data Systems Inc., in Westboro, Massachusetts, which makes the R&R Relational Report Writer for Dbase.

But even Henderson warned that "we can't deal with uncertainty. The nature of these preannouncements is really personally disturbing. If sales weren't doing so well, I might be singing a different tune."

Microsoft Ships

1988

What controls software reliability

- Bugs encountered and their impact
  - Functional/temporal bugs: focus here
  - Vulnerabilities: later
- Controls
  - Better software development
  - Thorough testing
  - Redundancy
  - Recovery

Basic Definitions

- Defect: requires a corrective action
- Defect density: defects per 1000 non-comment source lines (NC LOC).
- Failure intensity: rate at which failures are encountered during execution.
- MTTF (mean time to failure): inverse of failure intensity.

In this case mean is not taken over time, rather it is an ensemble average.

Basic Definitions (2)

Limited use in Software Reliability Engineering

- Reliability
  - R(t)=p{no failures in time (0,t)}
- Transaction reliability: probability that a single transaction will be executed correctly.
- Test Time: may be measures in CPU time or some measure of testing effort.

Why is Defect Density Important?

- Important measurement of reliability
- Often used as release criteria.
- Typical values of defect density /1000 LOC mentioned in literature:

Beginning Of Unit Testing	On Release		
	Frequently Cited in literature	Highly Tested programs	NASA Space Shuttle Software
16	2.0	0.33	0.1

Actual mean, median for 109 projects: 7.5, 4
Shah et al, 2012

- Long term trend: tolerable defect density limits have been gradually dropping, i.e. reliability expectations have risen.

Note: NASA space shuttle controversy: see appendix.

Static and Dynamic Modeling

- Reliability at release depends on
  - Initial number of defects (parameter)
  - Effectiveness of defect removal process (parameter)
  - Operating environment
- **Static modeling**: estimate parameters (defect density, defect finding rate*) before testing begins
  - Use static data like software size etc.
- **Dynamic modeling**: estimate parameters during testing
  - Record when defects are found etc.
  - *Time* or *coverage* based

* Partially unsolved problem

What factors control defect density?

- **Need to know for**
  - *static estimation of initial defect density*
  - *Finding room for process improvement*
- **Static defect density models**: The defect density is influenced by a number of factors f_1, f_2, etc. The models combine the impact of factors in two ways:
  - *Additive (ex: Takahashi-Kamayachi)*

    $D=a_1f_1+a_2f_2+a_3f_3\ldots$
  - *Multiplicative (ex. MIL-HDBK-217, COCOMO, RADC)*

    $D=C.F_1(f1).F_2(f_2).F_3(f_3)\ldots$

A Static Defect Density Model

- Li, Malaiya, Denton (93, 97)

$$D=C.F_{ph}.F_{pt}.F_{m}.F_{s}.F_{rv}$$

- *C is a constant of proportionality, based on prior data, used for calibration.*
- *Default value of each function* F_i *(submodel) is 1.*
- *Each function* F_i *is a function of some measure of the attribute.*
  - *The function sub-model needs to be determined using available data or reasoning.*

Possible factors

Phase

Programming Team

Process Maturity

Structure

Requirement Volatility (Code churn)

Submodel: Phase Factor F_{ph}

- The table shows possible values, based on numbers reported in the literature (Musa, Gaffney, Piwowarski et al.) for the **Waterfall development.**

At beginning of phase	Multiplier
Unit testing	4
Subsystem testing	2.5
System testing	1 (default)
Operation	0.35

- The values are to give you an idea of variability. Actual values will depend on specific process.
- Agile development: Cycles (Sprints) of 1-4 weeks, with daily 15 min scrum sessions for review. Research needed: Data, Analysis.

Submodel: Programming Team Factor F_{pt}

- Based on a study by Takahashi, Kamayachi, who found that defect density declines by about 14% per year (up to seven years).

Team's average skill level	**Multiplier**
High	0.4
Average	1 (default)
Low	2.5

- It is agreed that programming team skills have a significant impact. However measuring skill is hard and there are no good quantitative studies.

SEI- Capability Maturity Model

- Software Engineering Institute Capability Maturity Model (will use CMM for SEI-CMM)
- Begun in 1986 from SEI and Mitre
  - framework for government to assess contractors
- Based on
  - Statistical quality control (Deming's TQM, Juran)
  - Quality management (Crosby)
  - Feedback from industry and government

SEI Levels

SEI Level	Key Feature	How many organizations?
1.Initial	ad hoc	75%
2. Repeatable	basic management	15%
3. Defined	standardized	8%
4. Managed	quantitative control	1.5%
5. Optimizing	continuous improvement	Handful (0.5%)

Estimating software costs: bringing realism to estimating By Capers Jones, 2007

Submodel: Process Maturity Factor F_m

- Based on Jones, Keene, Motorola data.

SEI CMM Level	Multiplier
Level 1	1.5
Level 2	1 (default)
Level 3	0.4
Level 4	0.1
Level 5	0.05

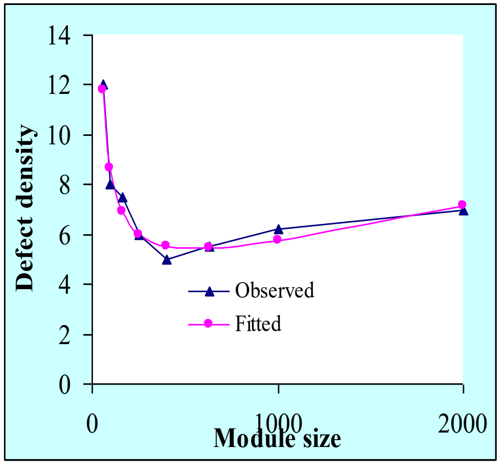
Submodel: Structure Factor F_s (Pt 1)

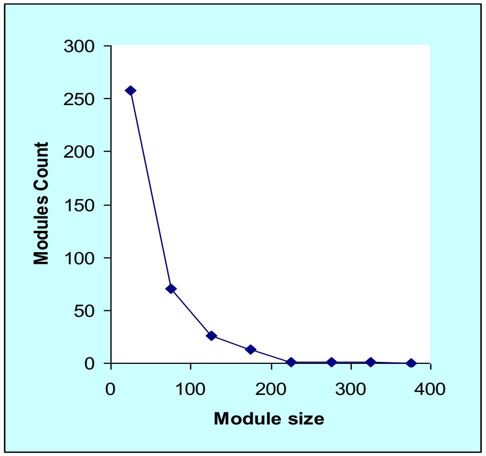
- Assembly code fraction: assuming assembly has 40% more defects
  - Factor=1+0.4×fraction in assembly
- Complexity: Complex modules are more fault prone, but there may be compensating factors, like people being more cautious when implementing them. No conclusive results are available that link measures like cyclomatic complexity with defect density.
- Note that by definition defect density is defects divided by software size, which itself is a complexity metric. Question is: does adding other complexity metric help?
  - Answer is that complexity metrics are strongly correlated with software size. Thus there is no compelling evidence.

Mamun, M.A.A., Berger, C. & Hansson, J. Effects of measurements on correlations of software code metrics. *Empir Software Eng* **24**, 2764–2818 (2019). https://doi.org/10.1007/s10664-019-09714-9.

Submodel: Structure Factor F_s (Pt 2)

- Module size: Data from several projects suggest that very small modules have higher defect densities (Fig 1). Note that many projects have a large number of small modules (Distribution in Fig 2)

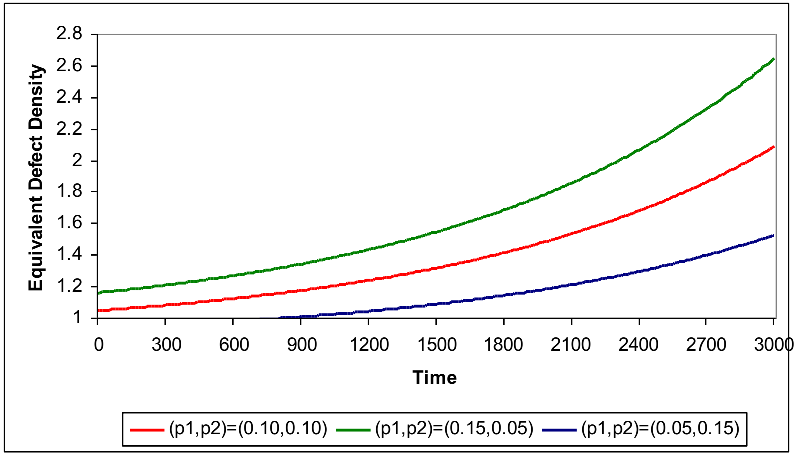


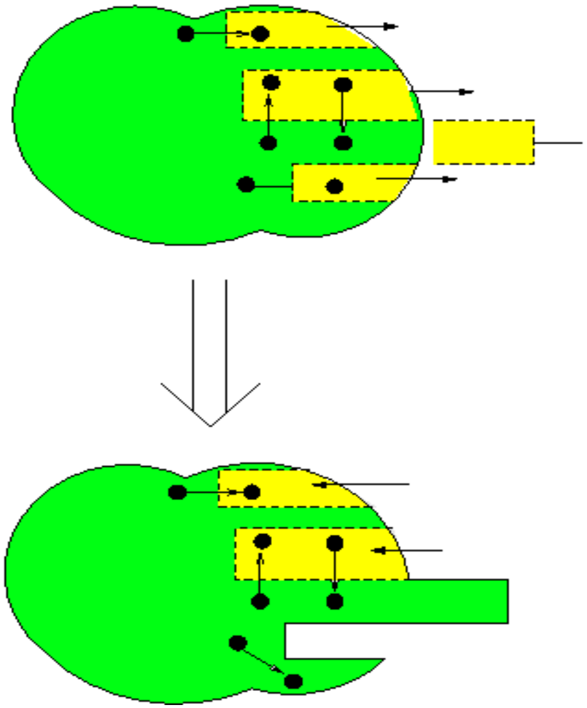


Y. K. Malaiya and J. Denton "Module Size Distribution and Defect Density," Proc. IEEE International Symposium on Software Reliability Engineering, Oct. 2000, pp. 62-71.

Submodel: Requirement volatility Factor F_{rv} (Code Churn)

- Impact depends on degree of changes and when they occur.
- Most impact when changes occur near the end of testing.
- Malaiya & Denton: ISSRE 99





Software Evolution

Successive versions of a program

- Bug fixes
- Added functionality
- Added support for new requirements

Results in

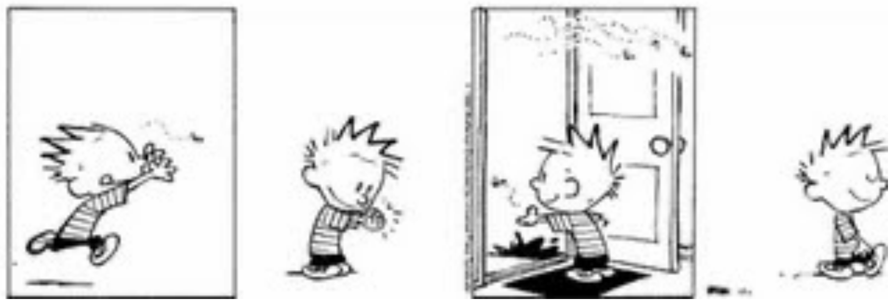
- Added code
- Modified code

Agile/DevOps

- Iterative development

Regression testing: after revisions

Regression:
"when you fix one bug, you introduce several newer bugs."



Reuse factor: A simple analysis

- u: fraction of software reused
- d_r, d_n: defect density of reused software, defect density of new software, $d_r < d_n$
- Total defects = $[u.\ d_r + (1-u).\ d_n]S$

  Where S is software size

- If there was no reuse, defects would be d_nS
- Normalizing,
  - Reuse factor $F_r(u, d_r/d_n) = [u.\ d_r / d_n + (1-u)]$
  - F_r is 1 if there is no reuse, <1 if reuse.

Using the Defect Density Model

- Calibrate submodels before use using data from a project as similar as possible.
- Constant C can range between 6-20 (Musa).
- Static models are very valuable, but high accuracy is not expected.
- Useful when dynamic test data (we will discuss this soon) yet available is not yet significant.

Static Model: Example

$D = C.F_{ph}.F_{pt}.F_m.F_s.F_{rv}$

•For an organization, C is between 12 and 16. The team has average skills and SEI maturity level is II. About 20% of code in assembly. Other factors are average (or *same as past projects*).

Estimate defect density at beginning of subsystem test phase.

•Upper estimate=16×2.5×1×1×(1+0.4 ×0.20)×1=43.2/KSLOC

•Lower estimate= 12×2.5×1×1×(1+0.4×0.20)×1=32.4/KLOC

Here the structure factor is 1+0.4×0.20 because of some assembly code. Factor 2.5 is for the beginning of the subsystem phase.

Static Models: Limitations

- Other multiplicative models like the COCOMO cost estimation model would have similar limitations.
- The parameter values are based on past projects, which may have been somewhat different.
- Calibration will be accurate only if data from somewhat similar projects was used.
- Some factors may be statistically correlated, for example Programming team and Capability Maturity factors.
- Still such models can be very useful at the beginning of projects for planning the test effort.

Vulnerability Density

- Vulnerabilities are defects that are security related.
- A fraction of defects are vulnerabilities.
  - What is that fraction?
  - To be discussed later.

- O. H. Alhazmi, Y. K. Malaiya , I. Ray, " Measuring, Analyzing and Predicting Security Vulnerabilities in Software Systems," Computers and Security Journal, Volume 26, Issue 3, May 2007, Pages 219-228.

- A. A. Younis and Y. K. Malaiya,"Relationship between Attack Surface and Vulnerability Density: A Case Study on Apache HTTP Server", ICOMP'12, 2012 Int. Conference on Internet Computing, July 2012, pp. 197-203.

What is lowest defect density achievable?

- What is the very best reliability level achievable, and how?

- Y. K. Malaiya, "Assessing Software Reliability Enhancement Achievable through Testing", Recent Advancements in Software Reliability Assurance 2019, pp. 107-138.

Appendix

Fact check

“Space-shuttle software - achieved a level of 0 defects in 500,000 lines of code”

NASA Space Shuttle Defect Density?

“Space-shuttle software - have achieved a level of 0 defects in 500,000 lines of code”   Widely quoted. Here is the source.

Code Complete: A Practical Handbook of Software Construction, Steve McConnel, Microsoft Press; 2nd edition (June 19, 2004)

- Industry Average: "about 15 - 50 errors per 1000 lines of delivered code."
- Microsoft Applications: "about 10 - 20 defects per 1000 lines of code during in-house testing, and 0.5 defect per KLOC (in released product (Moore 1992)”.
- "Harlan Mills pioneered 'cleanroom development', a technique that has been able to achieve rates as low as 3 defects per 1000 lines of code during in-house testing and 0.1 defect per 1000 lines of code in released product (Cobb and Mills 1990).
- A few projects - for example, the space-shuttle software - have achieved a level of 0 defects in 500,000 lines of code using a system of format development methods, peer reviews, and statistical testing."

NASA Space Shuttle Defect Density?

- Widely quoted, for example:

- R. V. Binder, "Can a Manufacturing Quality Model Work for Software?," IEEE Software, vol. 14, no. , pp. 101-102,105, 1997.
  - NASA Space Shuttle Avionics have a defect density of **0.1 failures/KLOC** (Edward Joyce, "Is Error-free Software Possible?" Datamation, Feb.18, 1989). Leading-edge software companies have a defect density of 0.2 failures/KLOC.

NASA Space Shuttle Defect Density?

- Nancy G. Leveson, Software and the Challenge of Flight Control, Chap 7 of Space Shuttle Legacy: How We Did It/What We Learned, 2013.
  - A mythology has arisen about the Shuttle software with claims being made about it being “perfect software” and “bug-free” or having “zero-defects”
- They Write the Right Stuff, Charles Fishman, Fast Company, 12.31.96
  - It is perfect, as perfect as human beings have achieved. Consider these stats : the last three versions of the program — **each 420,000 lines long-had just one error each**. The last 11 versions of this software had a total of 17 errors. Ten years ago the shuttle group was considered world-class. Since then, it has cut its own error rate by 90%.
- NASA Space Shutttle project lasted from April 12, 1981 to July 21, 2011
  - (30 years)

Meta-study: Open source vs Closed, Programming Language

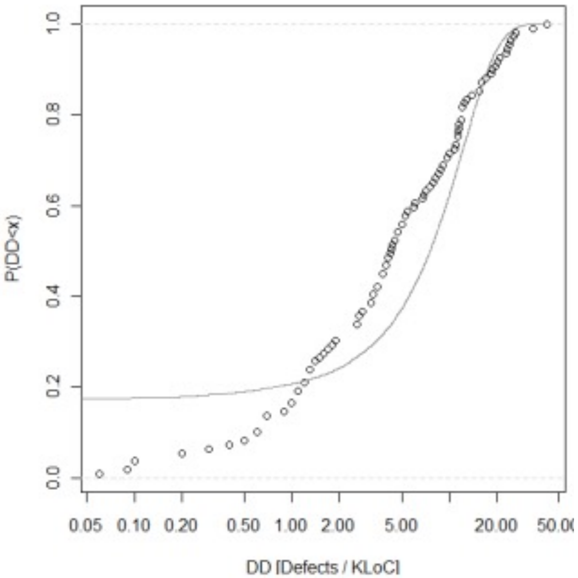


Figure 1. DD cumulative distribution

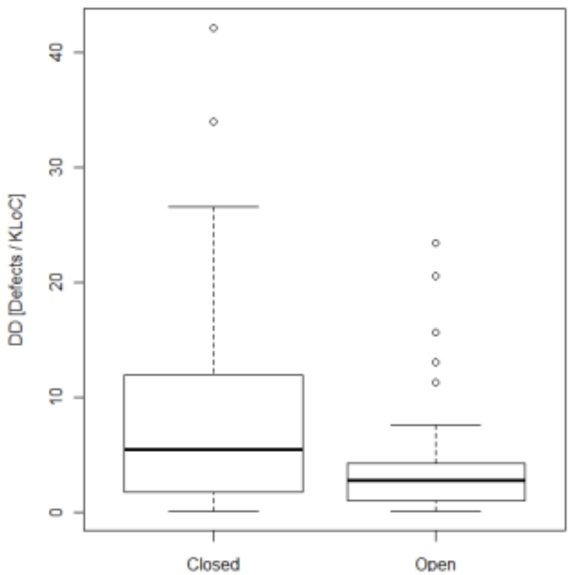


Figure 3. Box plot of Close Source vs. Open Source DD

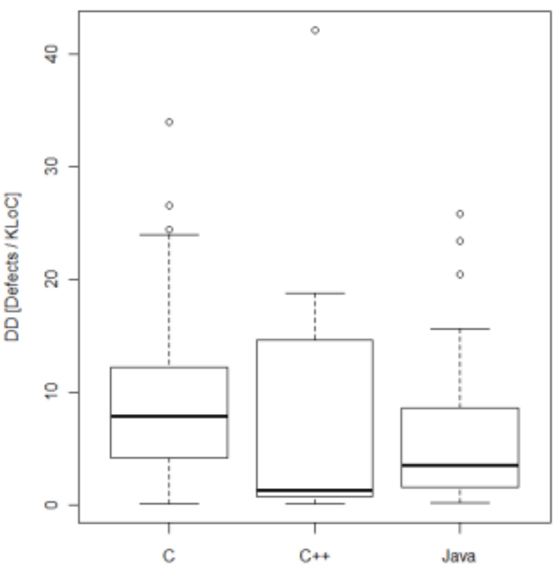


Figure 5. Box plot of DD per programming language

An Overview of Software Defect Density: A Scoping Study, 2012

Defect Density per function point

- Defect density per function point: 0.02- 0.1
- LOC/FP media values:
  - C: 99   (0.1/FP = 1/KLOC)
  - Java 53
  - Excel 191

QSM Function Points Languages Table
Defect Density Measurement, 2011

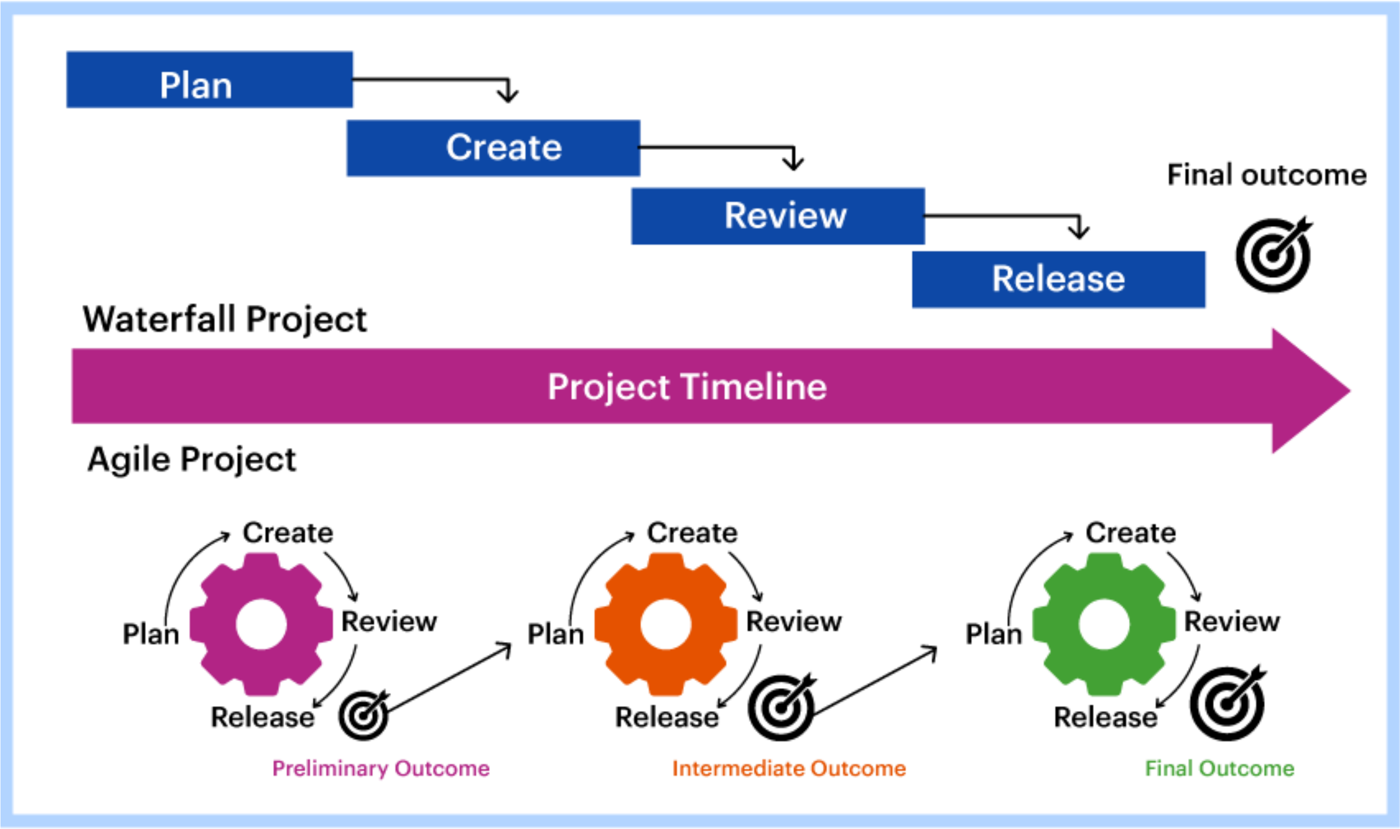
Problems in Software Testing research

- Assuming that there are typical/average faults
  - Assuming testability of the faults is normally distributed
  - Assuming injected faults represent actual faults
- Not recognizing multiple causes of defect density variations
- Excessive influence of research circles
  - Researchers in a circle citing each other, unaware of other existing observations/research. Reliance of circle specific jargon.

Appendix

Key ideas in Agile Development

Traditional vs Agile 2001 Project Management

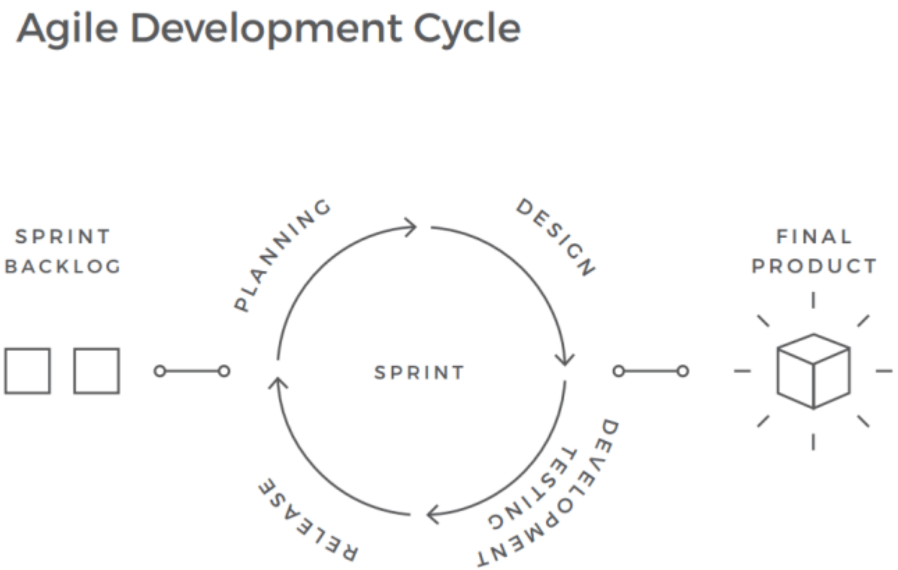


Scrum 1986 Framework Jargon

- Sprint: typically 1-4 weeks, sprint goal
  - Sprint Planning, development, Sprint review, Sprint retrospective
- User stories: informal, natural language description of features of a software system
- Product backlog prioritized list of functionality, Sprint backlog, Release backlogs Features need to be implemented for a release, velocity amount of work a team can deliver during a sprint
- Burndown chart work left to do versus time, Kanban board
- Scrum team (One product owner, scrum master, developers), Stake holders
- Daily Scrum meetings (standup) among team members
- Tools planning/management of work

The Scrum Guide, 2020, K Schwaber & J Sutherland

Agile software development cycle



For self study

The 4 Agile Values

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value -

1. **Individuals and interactions** over processes and tools
2. **Working software** over comprehensive documentation
3. **Customer collaboration** over contract negotiation
4. **Responding to change** over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

AGILE & SCRUM TIP SHEET

https://vitalitychicago.com/blog/agile-and-scrum-tip-sheet/

The 12 Agile Principles

1. Highest priority is to **satisfy the customer** through early and continuous delivery of solutions.
2. **Welcome changing requirements**, even late in development. Agile processes harness change for the customer's competitive advantage.
3. **Deliver working software frequently**, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Business people and developers must **work together** daily throughout the project.
5. Build projects around motivated individuals. Give them the environment and **support** they need, and **trust** them to get the job done.
6. The most efficient and effective method of conveying information to and within a development team is **face-to-face conversation**.
7. **Working software** is the primary measure of progress.
8. Agile processes promote **sustainable development**. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to **technical excellence** and good design enhances agility.
10. **Simplicity**–the art of maximizing the amount of work not done–is essential.
11. The best architectures, requirements, and designs emerge from **self-organizing teams**.
12. At regular intervals, the team **reflects** on how to become more effective, then tunes **and adjusts** its behavior accordingly.

The Scrum Roles

SCRUM TEAM – The Scrum Team is 10 or fewer people including one Product Owner, one Scrum Master and Developers.

- PRODUCT OWNER – A single decision-maker who is responsible for prioritizing the backlog and maximizing the value delivered by the Scrum Team. Interviews customers, reviews product feedback, analyzes market trends and works with upper management
- SCRUM MASTER – Servant Leader, coach & trainer who supports the Scrum Team, Product Owner and Org to adopt Scrum as defined in the Scrum Guide.
- DEVELOPERS – Cross-functional team of 3-9 people who plan, adapt and hold each other accountable to deliver a usable increment each sprint.

Events in the Scrum Framework

The Sprint – A timebox of one month or less which contains all the other Scrum Events. Every Sprint should be the **SAME LENGTH**.

- Sprint Planning by the scrum team–addresses the questions of WHY, WHAT and HOW and results in a Sprint Backlog which is a plan for the upcoming Sprint.
- Daily Scrum –is a short meeting for the Developers to synch their efforts, assess progress toward their sprint goal, and plan for the day ahead Also called Stand Up. Should last **LESS THAN 15 MINUTES** and be held at the same time and place every working day of the sprint.
- Increments of 'Done' work. DONE = Releasable
- Sprint Review –is an inspection of the actual working output of the sprint. It allows the team to demonstrate their progress, showcase the product and get feedback.
- Sprint Retrospective –is an event where the Development Team meets to discuss how to improve their process to make it more effective and enjoyable.
- Update Product Backlog

The Iteration/Sprint Cycle

- Iteration Planning Iteration is generalization of term sprint.
- Daily Stand Up a short, daily meeting to discuss progress and identify blockers
- Increments of 'Done' work DONE = Releasable
- Iteration Review /Demo
- Iteration Retrospective
- Update Product Backlog

https://agiletraining.com/wp-content/uploads/2014/09/AgileScrumCheatSheet-v.2014-09-15.pdf

Glossary

- STORY: a description of a small valuable customer requirement
- FEATURE/THEME: grouping of related stories
- BACKLOG: one list containing all stories
- STORY POINTS: a relative measure of complexity for a story.
- RELEASE PLAN: rough schedule of iterations
- RELEASE: Moving 'Done' stories to production
- BURN UP CHART: demonstrates visually how many points the team got 'Done’
- VELOCITY: how many points the team got 'Done' in an iteration.
- TEAM: cross-functional group working together to get a story 'Done'.
- TASKBOARD: where the team tracks their tasks visibly.
- SCRUM MASTER: owns the process, leads the team by empowering them, removes impediments, tracks progress.
- IMPEDIMENT: anything stopping progress on a task.

Agile and Scrum Cheat Sheet

Roles & Artifacts

Roles

- Product Owner- Develops product vision. Owns product backlog. Prioritizes backlog items
- Scrum Master: Process coach. Removes impediments. Facilitates team meetings.
- The Team: Cross-functional, Self-organizing, Completes all the work

ARTIFACTS

- Product Backlog: Lists all work on a product, Never complete, always changing, Higher order = higher priority
- Sprint Backlog: Lists work to be done in the current sprint. Pulled from product backlog. Items are broken into tasks
- Burndown Chart: Shows amount of work remaining/complete. Provides visual of current status. Shows likelihood of on-time completion

The beginner's Scrum CHEAT SHEET

ACTIVITIES

Sprint: Consistent iteration of time where the team completes work.

TIMEBOX: 1 week to 4 weeks (shorter for higher risk project. Default 2 weeks)

Sprint Planning: Plan created for what is to be delivered in the upcoming sprint.

FREQUENCY: END OF EACH SPRINT, TIMEBOX: 4 HOURS OR LESS

Daily Scrum (Stand Up): Daily 15 minute meeting where team mates report:

1.what did you do yesterday, 2. what are you doing today, 3. list any impediments

FREQUENCY: DAILY (AM), TIMEBOX: 15 MINUTES W/ MEET AFTER

Sprint Review: Team shows off completed work to the Product Owner

FREOUENCY: END OF EACH SPRINT, TIMEBOX: 1-2 HOURS

BACKLOG REFINEMENT: Systematically review each work item to clarify understanding, provide greater detail, and estimate the effort required.

FREQUENCY: AT LEAST ONCE PER SPRINT, TIMEBOX: 1 HOUR

Sprint Retrospective (Retro): Team discusses how the sprint went (except Product Owner)

ERFOUENCY: END OF EACH SPRINT, TIMeBox- 1 HOuR OR LESS

The beginner’s Scrum CHEAT SHEET, SCRUM Team Events