STEVENS
INSTITUTE OF TECHNOLOGY
1870

SSW533 Software Metrics and Estimation Guest Lecture: Performance Metrics

Dr. Andre B. Bondi

Revised March 2023

Road Map: Performance Metrics and Software Development Effort

Lecture: Performance (Dr. Bondi)

- Why Performance Metrics?
- Response Times and Latencies
- Device Utilizations
  - Visible through the Task Manager and perfmon on Windows systems, the Activity Monitor on Apple systems, *vmstat* on Linux and UNIX systems.
- Effort and improvements of fixing performance issues (Prof. Xiao)

Why are we interested in Performance Metrics?

- Interactive systems: short response times are pleasing
  - Long response times can cause business to be lost
- Batch systems: deadlines constrain execution time (e.g., payroll, transaction clearance, bank statements, system backup)
- Industrial control and monitoring systems:
  - Response time and throughput requirements imposed by engineering needs
  - Slow systems can be dangerous
  - Communicating systems of any kind impose latency constraints
- All systems have throughput requirements (domain specific)
- Poor performance can yield insights about hidden malfunctions
- Capacity planning, resource utilizations

NOTE: different response time requirements and throughput requirements, memory requirements for different applications

Examples of poorly performing systems

- US http://healthcare.gov was slow on rollout; malfunctions revealed
- Erratic response times or resource usage under constant load could indicate concurrency problems
- Taylor Swift fiasco at Ticketmaster in 2023
- NJ covid vaccination and testing websites crashed on overload
- NJ unemployment benefits website malfunctioned and ran slowly on overload during pandemic
- Brexit referendum voter registration website crashed on day of deadline to register
- Many more newsworthy undesirable events like these

More uses of performance metrics

- Diagnosis of functional problems in concurrent and distributed systems
  - Deadlock
  - Data corruption because mutual exclusion does not work properly; causes errors or repeated execution attempts

When do we think about performance measurement?

Run time:

- Workload, offered load
- Response time, throughput, resource utilizations, bandwidth usage
- Behavior under steady, constant load
- Behavior under burst loads
- Performance during stressful events or failures
- AFFECTS CLOUD COST

During the SDLC:

- Design measurements into the system architecture
- Tracking return on investment (ROI) from improving performance
- Reporting and resolving performance issues and bugs
- Poor performance is a frequent cause of software project cancellation

Economic cost of poor performance

- A frequent cause of software project cancellation
- Dissatisfied, lost customers
- Delayed unemployment benefits
- Delayed vaccinations hurt public health
- Delayed software release until performance issues observers in testing are fixed
- Poor performance triggers need for software rework
- Performance problems can be hard to debug, even if they are not hard to observe
- Competitive differentiator: Firefox slower than Chrome could be increasing Chrome's market share
- Cloud costs: a huge expense!

Performance Metrics

- System and Software performance metrics are gathered and evolve over time
- Externally focused … e.g., observable by a user or outside of the system
  - To answer the question (remember GQM?)
    - How well is the system performing?
    - Can it handle spikes in load?
    - Are there any trends, such as increasing response time, increasing load?
    - How do the traffic and performance metrics vary over time?
  - Internally focused:
    - Host utilizations
    - Code hooks and instrumentation
    - Observability (metrics collection from top to bottom)

Performance metrics to consider

- Will look at:
  - Latency (Response Time):
    - Average, median, variance,
    - percentiles, max/min observer in a time interval
    - Trends and variation by season, time of day
  - Achievable system throughput (offered load, processed load)
  - If a system is unavailable, achievable throughput is ZERO.
- Also of interest, but not much published about it:
  - Execution duration of builds, test suites, configuration changes, provisioning, etc.
    - Long build time after changing a few lines of code, regression test
  - DevOps latency
  - ❖ These are open research issues.

Sources of Performance Data

- Run time:
  - Response times from load generation software (end time – start time)
  - Resource utilizations from operating system instrumentation or software to obtain it from the operating system
  - Hooks within code, DBs, or applications (can take the form of APIs)
  - Applications to monitor trends, generate alarms, anomaly detection
  - Observability
- SDLC:
  - Tracking of performance issues in bug tracking logs or collaboration tools (e.g., Slack, JIRA)
  - Code profiling
  - New measurement tools and measures coming on stream every year
    - Evolving growth area

Code and Execution Profiling

- Used to measure where the code is spending its time
- Numbers of loop iterations
- Numbers of times each function is called
- Numbers of API invocations
- Numbers of executions of code blocks
  - If {if block}, Else {else block} , Elseif {elseif block}
- Hooks in code to measure time spent on each iteration, response times of function or API calls

Since 2010, many commercial tools and open-source have appeared on the market to gather, display, and analyze performance data

How do we typically analyze performance?

- Performance Measurement
  - Direct Observation of System (only possible once built and under test or in service)
- Performance Prediction
  - with parameters from measurement data, estimates, or past experience
  - Analytical Performance Modeling
    - Mathematical Models
  - Simulation Modeling
    - Specialized Tools – based upon characterization of components

Analyzing Performance Test Results

- Interpretation depends on test design
- Can be done with reference to simple models
- Always wish to track deviations from baseline after changing code, or configuration, just as one needs a functional baseline when testing functionality

Analyzing any performance measurements

- Averages and other statistical quantities should only be taken over time periods in which the system is stable and has a constant load
- Trends and high load variability during the observation time can make statistics meaningless or misleading
- Always watch out for oscillations and trends
- Mathematical reason:
  - Concept of stochastic equilibrium (outside the scope of this course)
  - Sampling statistics should be collected over homogeneous statistical populations
  - Variation should be examined separately

Performance is a vast field

- Interdisciplinary
- Need knowledge of
  - Systems, protocols, operating systems, computer architectures
  - Domain-specific performance requirements
- Analysis methods:
  - Operations research
  - Statistics
  - Stochastic models
  - Simulations
- Approach must be tailored to the circumstances at hand
- Always be rigorous about assumptions and data collection

Reading Material

Software Performance Engineering:

- International Conference on Performance Engineering: https://icpe.spec.org/
  - *Accessible via the ACM digital library*
- Bondi, André B. *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice*. Addison Wesley, 2014.
  - *This is in the Stevens library.*

Return on Performance Intervention:

- Y. Zhao, L. Xiao, A. B. Bondi, B. Chen and Y. Liu, "A Large-Scale Empirical Study of Real-Life Performance Issues in Open Source Projects," in *IEEE Transactions on Software Engineering*, vol. 49, no. 2, pp. 924-946, 1 Feb. 2023, doi: 10.1109/TSE.2022.3167628.
  - *Accessible via the Stevens library and IEEE Explore*

STEVENS
INSTITUTE OF TECHNOLOGY
1870

Part 2: Analysis of Real-World Performance Issue Reports

Presenter: Lu Xiao
Associate Professor
Department of Systems Engineering

lxiao6@stevens.edu

Revised 10/27/2025

What is the Effort and Benefits of Fixing Performance Issues?

- Software development always faces time pressure and limited human resources. It is important to focus on the cost-effectiveness in the development activity

- Zhao et al's study [1] focuses on analyzing the fixing of performance issues through mining software repositories. In this study they measure:
  - The performance improvement achieved on fixing performance issues
  - The effort on fixing the performance issues
  - Other aspects of considerations mentioned when fixing the issues

An Example Issue Report

PDFBox-591: PDFBox Performance Issue: BaseParser.readUntilEndStream() rewrite
Root Cause: The current implementation of this method uses a very slow test for end of stream conditions. A profile of the readUntilEndStream() method shows that a huge chunk of the method's processing time is being consumed in the cmpCircularBuffer() call - which is purely part of the test for the end of stream marker. In other words, the readUntilEndOfStream() is spending twice as much time testing for the end of stream marker as it is reading bytes from the stream.
Proposed Solution: A better solution is to use a simpler, direct fail-fast test conditional structure that uses byte primitives. I strongly recommend that the current method be removed and replaced with the following code below.
Profiling Data: This results in a relative speed up of readUntilEndStream() method of a little over a factor of 3 (a ratio of 113/37 = 3.05 if you want to be more precise). This in turn helps the overall performance of PDDocument.parse() by about a factor of 2.7.
Other Concerns: Note the addition of some byte constants used to make the code readable."

Fig. 3: Issue Annotation-PDFBOX-591

https://issues.apache.org/jira/browse/PDFBOX-591

- The description talks about:
- How the issue was caused
- How the issue should be fixed
- What is the achieved improvement (e.g. factor of 3)
- Other considerations besides performance, i.e. readability in this example

Dataset

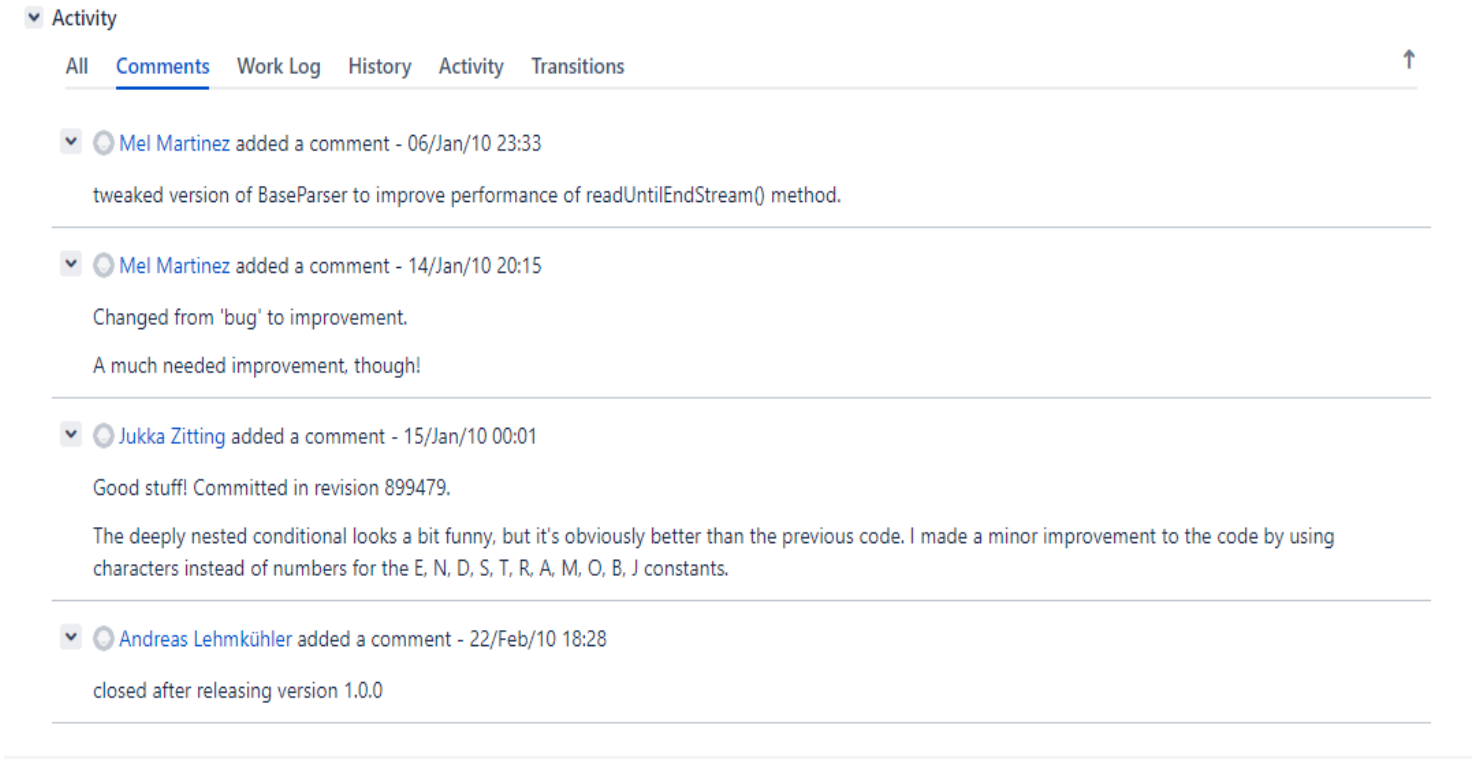
- A total of 570 performance issues
- Selected from 13 open source projects from Apache Software Foundation
- The projects are implemented in three languages, Java, Python, C/C++
  - 192 performance issues from Java projects
  - 162 performance issues from Python projects
  - 216 performance issues from C++ projects
- The projects are in different problem domains such as database, distributed systems, programming language, dependency managers, etc.

  https://projects.apache.org/projects.html?category.

Analysis Methodology---Effort

- Estimate the **effort** for fixing performance issues based on two proxies:
  - Number of developers who participated in the discussion of the issue report. Generally, more developers involved, the more difficult/expensive it is to address the issue.
  - Number of discussion comments associated with an issue report. More discussions are needed for addressing an issue, it is more difficult/expensive to address.

An Example Issue Report for "Effort"



Activity

All Comments Work Log History Activity Transitions

Mel Martinez added a comment - 06/Jan/10 23:33

tweaked version of BaseParser to improve performance of readUntilEndStream() method.

Mel Martinez added a comment - 14/Jan/10 20:15

Changed from 'bug' to improvement.

A much needed improvement, though!

Jukka Zitting added a comment - 15/Jan/10 00:01

Good stuff! Committed in revision 899479.

The deeply nested conditional looks a bit funny, but it's obviously better than the previous code. I made a minor improvement to the code by using characters instead of numbers for the E, N, D, S, T, R, A, M, O, B, J constants.

Andreas Lehmkühler added a comment - 22/Feb/10 18:28

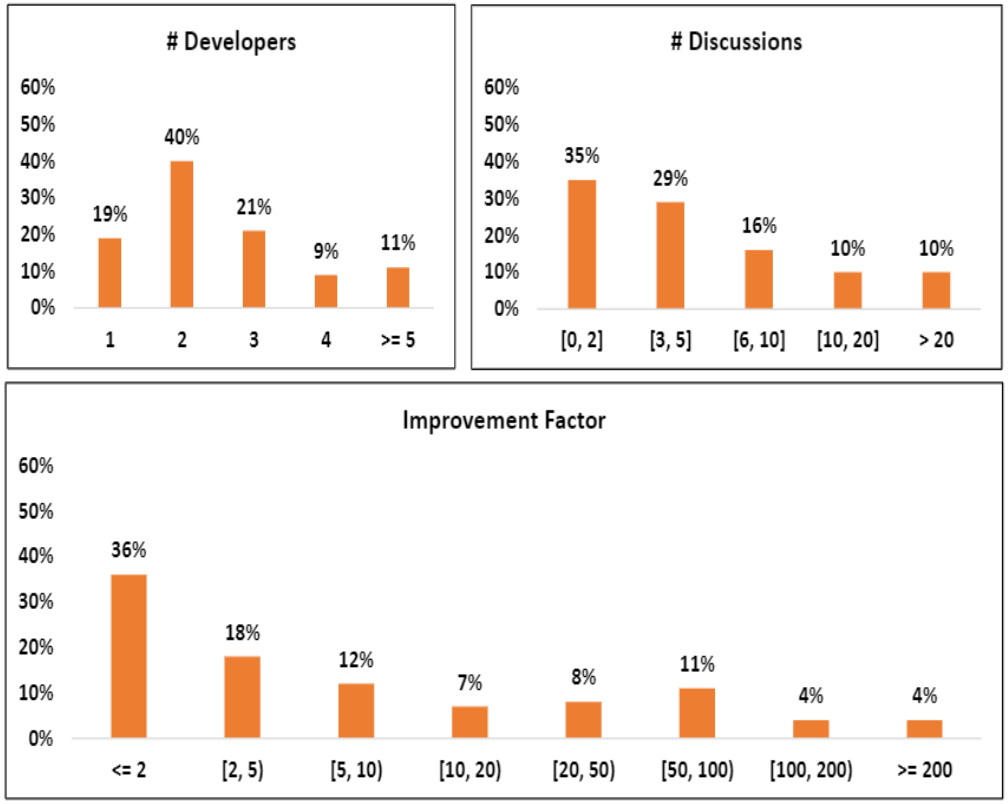
closed after releasing version 1.0.0

https://issues.apache.org/jira/browse/PDFBOX-591

Analysis Methodology--- Effectiveness

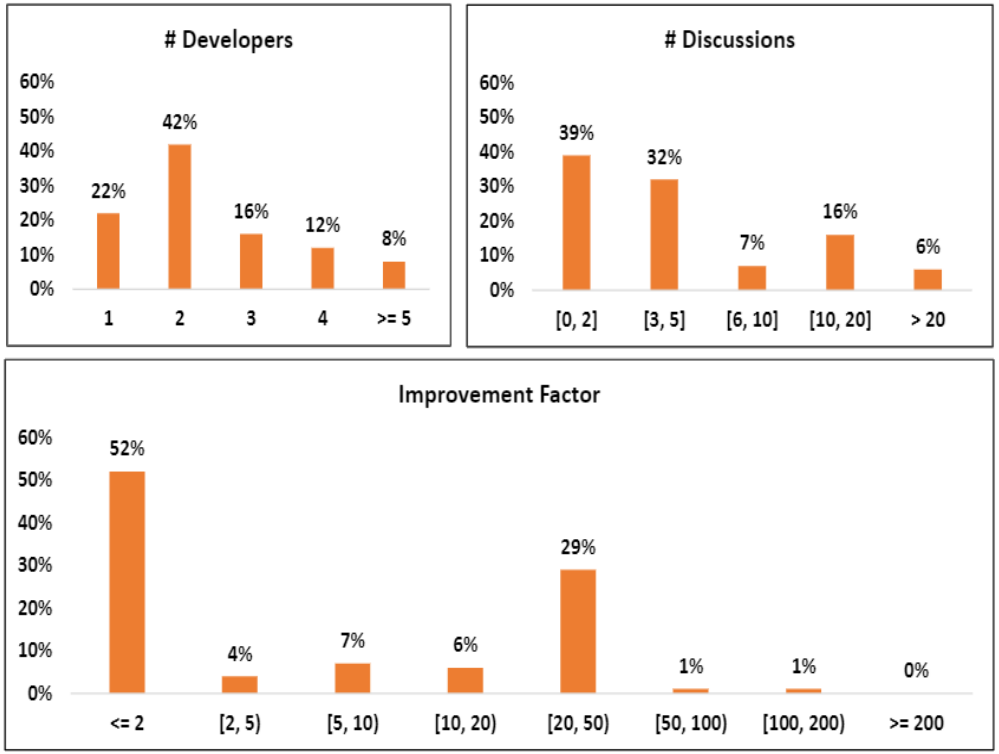
- Estimate the effectiveness in two aspects:
  - **Improvement Factor**, which measures the extent of performance improvement after the issue is being fixed based on the profiling data. This could be based on different metrics, discussed earlier, such as throughput, response time, or others.
  - **Other aspects of concerns**, such as code readability and maintainability.
    - For example, in Issue PDFBox-591, "*Note the addition of some byte constants used to make the code more readable*."
  - *Note that not every issue report contains information that can derive the "effectiveness"!!*

Cost-Effectiveness (Java Projects)



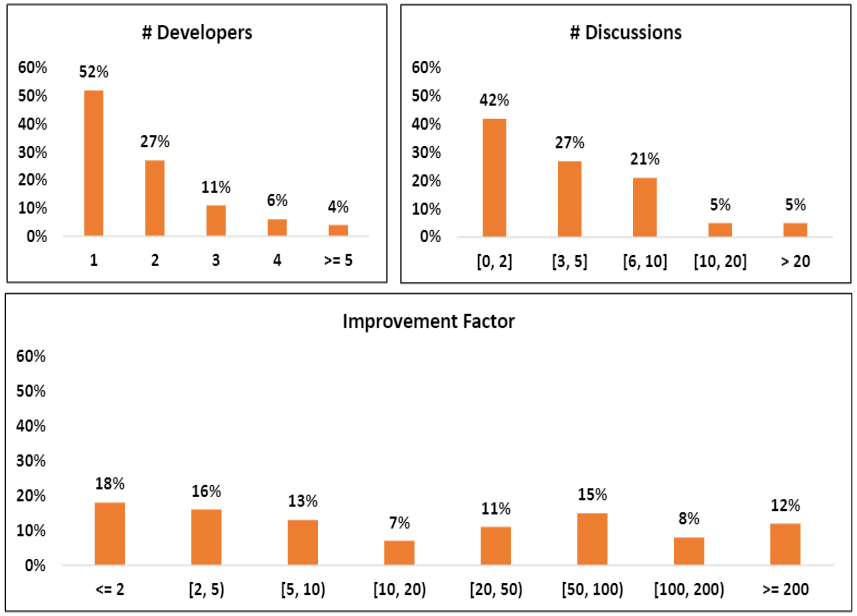
- Most issues are addressed by fewer than 2 developers and have fewer than 5 comments
- While the "effectiveness" can vary in a large range up to more than 200 times!!!

Cost-Effectiveness (Python Projects)



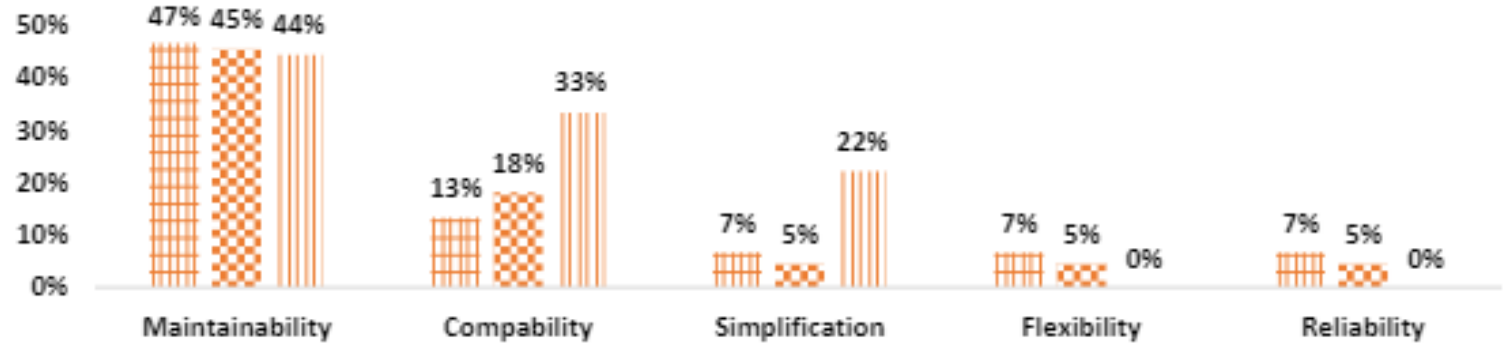
- Most issues are addressed by less than 2 developers and have less than 5 comments
- 52% of Python issue yield to less than 2x performance improvements, but 29% of Python issues have 20 to 50 times performance improvement.

Cost-Effectiveness (C++ Projects)



- Most issues are addressed by less than 2 developers and have less than 5 comments.
- 35% C++ performance issues yield to more than 50 times performance improvement, and 12% issues even reach more than 200 improvement after fixing.

Other Concerns



- A total of 46 performance issues (8%) also involve other concerns in seven aspects—maintainability, compatibility, readability, security, flexibility, simplification, and reliability.
- Of a particular note, the influence on the other aspects of concern is not always positive in performance optimization. That is, the resolutions of 85% of the 46 issues yield positive influence on these other concerns; while for the remaining 15% issues, developers trade off other concerns for performance optimization.

Take Away Messages

- Regardless of the programming language, it takes less than 2 developers and less than 5 discussions to resolve a performance issue in most cases.

- We did not observe statistically significant impact from programming language on the “effectiveness”. However, it is worth noting that C++ has the highest potential performance improvement — 12% of the issues yield to more than 200 times improvement.

- Furthermore, in 8% performance issues, developers also concern about seven other aspects of quality attributes when fixing the performance issues — maintainability is the most common concern in addressing performance issues.

Assignment for you: Understanding Real World Performance Issues

- **Goal:** Understand how real software engineers identify and solve performance problems in large systems.
  You'll examine **real Jira issue tickets** from the Apache Avro project.
- **What you will do:**
  - Review 3 real issue reports
  - Decide whether each involves performance
  - If yes → analyze causes, improvements, and effort
  - Reflect on what you learned about debugging performance issues
- **Total Time:** ~1 hour
  This is **individual work**

Why Are We Doing This?

- Performance problems in real systems are **messy** — rarely one-line fixes
- Engineers must:
  - Interpret incomplete or vague issue descriptions
  - Balance performance with maintainability, readability, and effort
  - Communicate findings clearly to teammates
- **This activity helps you:**
  - ✅ Learn *how* performance problems are discovered
  - ✅ Practice *reasoning* about trade-offs
  - ✅ Build skill in *reading real-world technical discussions*

After You Finish the Task

- After submitting your assignment, we invite you to complete two **very short and voluntary** steps:
  - Survey (5 min): Tell us about your experience with this assignment
  - Consent Form (1 min): Allow your anonymized work to be used in improving course design and teaching research
- Your grade is NOT affected by whether you complete these
- Your identify is not collected---results are anonymous
- Participation simply helps us make the module better