Experience with Teaching Performance Measurement and Testing in a Course on Functional Testing

Andre B. Bondi*
Software Performance and Scalability Consulting LLC
Red Bank, NJ
http://andrebbondi.com
andrebbondi@gmail.com

Razieh Saremi
Stevens Institute of Technology
Hoboken, NJ
rsaremi@stevens.edu

ABSTRACT

Stevens Institute of Technology offers a graduate course on functional software testing that addresses test planning driven by use cases, the use of software tools, and the derivation of test cases to achieve coverage with minimal effort. The course also contains material on performance testing. Teaching performance testing and measurement in a university setting was challenging because giving the students access to a target system would have required more time, resources, and planning than were available. We addressed these challenges (a) by showing the students how resource usage could be measured in a controlled way with the instrumentation that comes with most modern laptops by default, and (b) by having the students use JMeter to measure the response times of existing websites . We describe how students were introduced to the concept of a controlled performance test by playing recordings of the same musical piece with and without video. We make recommendations for the future avoidance of the emergent ethical issue that one should not subject one does not own to anything but the most trivial loads. We also describe some successes and pitfalls in this effort.

CCS CONCEPTS

• **Software and its engineering → Software performance**; Software testing and debugging.

KEYWORDS

Performance testing, performance measurement, software testing, education

ACM Reference Format:
Andre B. Bondi and Razieh Saremi. 2021. Experience with Teaching Performance Measurement and Testing in a Course on Functional Testing. In *Companion of the 2021 ACM/SPEC International Conference on Performance Engineering (ICPE '21 Companion), April 19–23, 2021, Virtual Event, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3447545.3451196

*Also with Stevens Institute of Technology.

ICPE '21 Companion, April 19–23, 2021, Virtual Event, France

ACM ISBN 978-1-4503-8331-8/21/04…$15.00
https://doi.org/10.1145/3447545.3451196

1 INTRODUCTION

The market for performance testing has grown rapidly in recent years[1]. Large-scale performance failures of both public- and private-sector websites underscore the need for it. Long response times on e-commerce sites are known to adversely affect revenue. While the need for performance is mentioned in at least one software architecture textbook[3], performance engineering *per se* is not widely taught in computer science and software engineering courses.

The authors have taught a course on functional testing in the graduate software engineering program at Stevens Institute of Technology. They recognized an opportunity to include material on performance testing that would expose the students to resource usage and response time measurement. Response time measurement had been included in past renditions of the course.

At Stevens, students are taught to use functional requirements, scenarios, and use cases to develop functional test plans and test cases. We built on this and on the material in [5] to show students how to use workload scenarios, performance requirements, and basic performance modeling to build performance test plans. At the same time, we wished to give the students practical experience of taking resource usage measurements under controlled conditions, as well as experience of measuring web page response times directly. Students' practical work was conducted on their own laptops using tools that are part of the operating system as well as tools that can be downloaded and used free of charge. Adding performance topics to a course on functional testing is consistent with the first author's view that performance should receive attention in all phases of the software development life cycle [5]. Stevens does not currently offer a course in performance engineering.

The purpose of this paper is to describe our experience with adding performance testing to a course on functional testing, and to make recommendations on how this addition could be improved. After describing the functional testing course and the place of performance testing within it, we shall discuss the setting of the performance context within the course. We then describe how we introduced students to resource usage measurement and response time measurement using tools that are freely available. We then address ethical issues concerned with sending test loads to websites one does not own, and make recommendations about setting up performance testing targets on which students can practice response time measurement. We also make recommendations about how much class time and lab time should be allowed for performance testing in this type of course.

2 OVERVIEW OF THE TESTING COURSE AT STEVENS

The Stevens course on testing is entitled *Software Testing, Quality, and Maintenance*. It includes an in-depth study of software testing methods throughout the development process. Students learn to effectively analyze and test software systems using a variety of methods and tools as well as to select the best-fit test case design methodologies to achieve the most suitable test strategy with highest possible test coverage. There is extensive hands-on project work with (a) an emphasis on testing as a technical investigation of a product undertaken to determine the system quality and (b) an emphasis on the clear communication of test results. The combination of lectures, readings, quizzes, and homework assignments, some of which require programming in Python, and some using open source tools are intended to provide the students with such capabilities as:

- Developing effective test plans and test strategies throughout the system life cycle,
- Selecting best fit test case design methodologies Based on test strategies,
- Preventing system defects and failures through appropriate early defect removal techniques such requirements reviews, code reviews and integration test,
- Designing and implementing tests and measure the results and the progress,
- Managing configurations and trouble reports to project the expected reliability and quality of a release,
- Understanding and testing for limited security vulnerabilities,limited automated testing tools, and limitations of verifying the correct behavior of concurrent programs,
- Being able to devise rudimentary performance tests from workload scenarios and performance requirements, and
- Being able to understand rudimentary resource usage measurements and the insights they can offer into how a software product works.

Resource usage measurement, performance requirements, and basic modeling were included in the Stevens course for the first time in the Spring 2020 semester. They were based in part on material in [5]. Much of the material on the functional testing part of the course is covered in a well-known textbook [11]. The assignments and examples cover novel and rapidly-developing research fields with industrial applications. The assigned course literature readings, included research papers chiefly from the last five years, for example, [4][11][10]. Some of the course projects have been submitted for publication [1,2,3].

The first author had covered resource usage measurement and its relationship to basic operational laws [6] while giving guest lectures in a course on software quality assurance taught by Prof. V. Cortellessa at the University of L'Aquila in 2016. Teaching both performance topics in the testing course at Stevens presented an opportunity to position performance requirements engineering and basic performance modeling as bases for performance test planning. Presenting performance testing at L'Aquila as part of a quality assurance course of which performance modeling and basic queuing modeling were major components provided an opportunity to bring the modeling concepts to life and show that they can be used to predict system behavior. This made the performance testing and performance modeling components of the L'Aquila course mutually reinforcing.

[1] D.Bobadilla, J. Adames, M.Hassany, V.Singh, "How Does the Student Class Exchange Website at Stevens Perform?" , Submitted
[2] J.Bangur, E.Buczek, T.Bhoir, J.Buonocore, "Static Test Analysis on Scrooge Mcduck's Bank Performace, a Case Study", Submitted
[3] Z.Lin, P.Li, S.Kamath, A.Bezzam, "Stevens Duck Home Website Usability Testing", Submitted

3 SETTING THE PERFORMANCE TESTING CONTEXT

We explain to our students that performance testing is best done only after functional tests of the use cases to be tested have been passed. We also note that performance testing may expose concurrent programming issues such as deadlock and violations of mutual exclusion that may not have emerged during functional unit tests. Furthermore, the output of a system that has had a performance test must be validated even though the system may have already passed functional tests. We briefly explain how to develop performance test cases with reference to workload specifications and performance requirements. We show how performance requirements must be cast in terms of the measurements that will be captured during testing. Performance test plans and scripts are developed based on the workloads and requirements. In the absence of a development project, it is not possible to have students test an entire system, but we can at least introduce them to the concepts and engage in a small amount of hands-on measurement.

4 EXERCISES IN PERFORMANCE MEASUREMENT

4.1 Using Native Resource Usage Measurement

The Windows task manager and Apple activity monitor both collect and graphically display data about hardware resource usage, globally and per process. One can use either of them to gain insights into what some applications might be doing on one's own computer. We asked the students play specific videos on YouTube while observing resource usage visually. They collected screen shots of the monitors' plots whenever they took specific actions, such as starting the video, stopping it, and when specific events occurred within the video. The purpose of these experiments was to show that different activities, or in this case, videos with different characteristics, can make different demands for resource usage.

- At both Stevens and L'Aquila, we had the students turn on their resource usage monitors and then run a video recording of the *Triumphal March* from Verdi's opera *Aida* with full staging of an orchestra and a moving chorus [14] and then by a wind band with neither video nor chorus [2]. The aim was to compare the behaviors of the resource usage measures observed over time, and to prompt students to use the measurements to make educated guesses about how the video application works. Streaming an opera video with full staging is more likely to use more resources, including the GPU, and to require the client-side receipt of more packets than streaming a recording that has an orchestra but no singers or video.

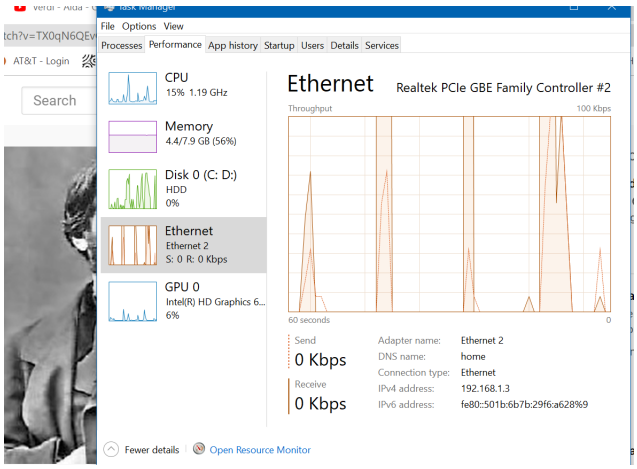

Figure 1: Windows performance monitor, Aida chorus with no video

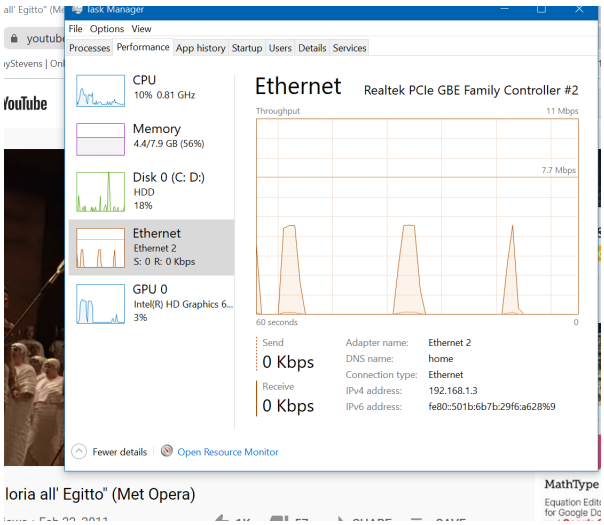

Figure 2: Windows performance monitor, Aida chorus with video

- In the spring of 2020, much of the testing course, including the lectures on performance, was taught over Zoom because of the COVID19 pandemic. The first author asked his section of the class to all turn off their cameras, and then to turn them all on at once after he counted to three. The instructor's instance of the Task Manager's performance monitor was visible to students via a shared screen. It indicated hardly any change in resource usage, except for a short spike in the incoming packet rate. We had expected a spike in GPU activity on the instructor's computer as well, but did not observe one.
- Students at L'Aquila were asked to view an excerpt from a recording of the 2006 World Cup soccer match between France and Italy and comment on what was measured when a goal was scored. Scoring a goal did not impact resource usage much. This video was not accessible in the USA, and so could not be used at Stevens.

Figure 1 and Figure 2 were generated by the first author on his Windows laptop. Without video, the peak incoming network usage while playing the *Aida* excerpt is 100 Kbps a minute or so into the recording, as shown in Figure 1. With video, the peak incoming network usage is about 5 Mbps, as shown in Figure 2.

Outbound network usage, as indicated by the dotted lines, is much lower than incoming network usage. With or without active video, the plot of network bandwidth usage forms a series of humps or peaks separated by periods of almost no activity. This suggests that the recording is being prefetched and buffered. With music alone and no video, the network bandwidth usage is much lower, and also has the hump pattern. The results might be different in successive runs if the same recording were to be played over and over, because of caching. In a future experiment, it could be interesting to put a packet sniffer on the user's machine to look at network traffic in detail. That was outside the scope of our classes and is also outside the scope of this paper.

4.2 Using JMeter to Measure Web Page Response Times

We asked students to identify a system of their choice with multiple users and generated page visits using JMeter[4]. We had the students design a workload testing scenario and execute it with Apache JMeter for a typical hour of a typical day on a live production system. They were required to use different metrics provided by JMeter such as latency and response time to analyze the performance of the system. Students were expected to provide insights based on throughput with respect to number of the users and received and sent bytes. The sent bytes are a surrogate for the page hit rate. The received bytes are determined by the amount of data required to render each returned page. Students were also supposed to use the generated data of the test plan reported in the standard JMeter reporting tool and provide deeper performance analysis insights for further test and strategy planning. To have a better understanding of JMeter metrics we briefly review some JMeter terminology [5].

- **Ramp up period** is defined as the time required to reach the peak load in the test,
- A **Thread Group** is a group of virtual users executing the same sequence of page visits,
- **Loop Count** is the number of times a Thread Group will execute,
- **Throughput** is calculated as the ratio of requests per unit of time. The time is calculated from the start of the first sample to the end of the last sample. This includes any intervals between samples, as it is supposed to represent the load on the server. The formula is:

$$Throughput = \frac{\#requests}{total\ time}$$

- **Load Time** is the duration of the test to be finished
- **Connect Time** is the time taken to establish connection with the server.

In class, the second author asked the students to try JMeter on a Stevens-owned website for different loads with 100 and 500 virtual users, and then re-run the test for two loops of 500 users

[4]https://jmeter.apache.org/index.html
[5]https://www.softwaretestinghelp.com/jmeter-components/

following the workload model in Table 1. While JMeter provides the option of scheduling threads arrival, we followed the basic JMeter planning. Users arrived one after the other. During this exercise, all the students present in the class tested the workload model on JMeter individually and reported the result. As summarized in Table 1, the virtual users in scenario 1 and scenario 2 arrived in the period of 1 second while the user load in scenario 2 released in the period of 2 seconds, one second per loop of users.

Table 1: Workload Model for Website Homepage

Metrics	Scenario 1	Scenario 2	Scenario 3
Total User #	100	500	500
Ramp Up Period (Sec)	1	1	1
Loop Count(Sec)	1	1	2
Throughput(Sec)	100	100	200
Load Time(Sec)	200	400	600
Connect Time(Sec)	600	1000	1500

Figure 3: Result of performance testing under Scenario 1, 100 users

Figure 3, Figure 4, and Figure 5 were generated by the second author on her laptop based on the workload specified in Table 1. They correspond to what a single student might obtain when running the scripts alone. As shown in Figure 3, the test was successfully completed. The standard deviation of the response time was 18 ms the average of 143 ms in the system. The throughput for this test was 5123.8 packets/min. The second scenario with 500 users, Figure 4, finished with a successful test. The inbound packet throughput increased to 3477.9 packets/min and decreased after that. the average response time is 2506 ms with standard deviation of 1318 ms.

It should be noted that these statistics were calculated during a period that included the ramp-up of the test from the test start time. Therefore, they do not reflect any information about the performance of the system in steady state. Computing the statistics from test startup ("time zero") is the default option in more than one load testing package, whether commercial or open source. Students and beginning practitioners should be reminded that the average measurements gathered during ramp-up and ramp-down period are not indicative of performance when the system is in steady state, as they will skew the values of the statistics.

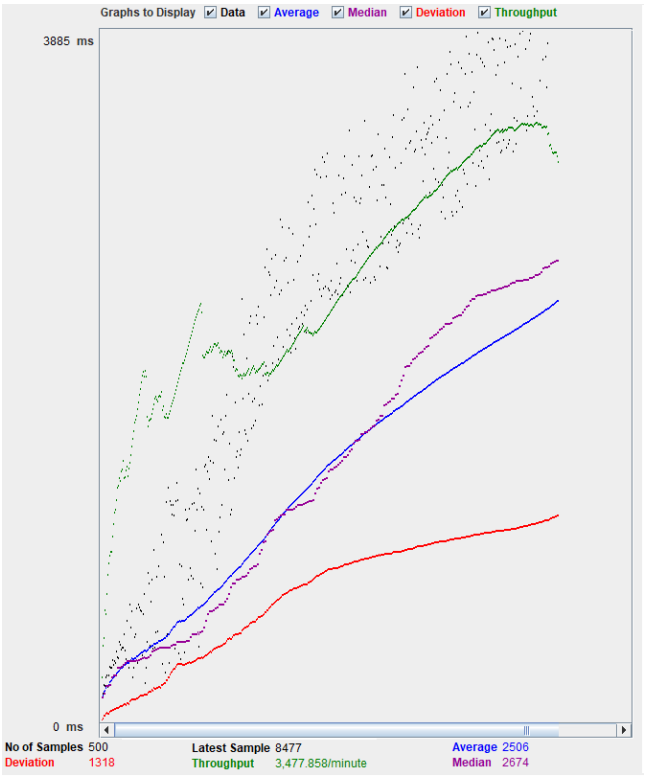

Figure 4: Result of performance testing under scenario 2, 500 users

During the third scenario the throughput was 383.51 packets/min and the response time standard deviation increased to 34185 ms with average response time of 41541 ms. Interestingly, when the second author ran the third scenario before the class exercises, it successfully passed the performance test with a throughput of 754.8 packets/min, and average response time of 3318ms, with a standard deviation of 3752ms. This is shown in Figure 5.

In the last test attempt with 500 virtual users, tests successfully ran during the first loop with 500 virtual users, but during the

second loop, the load generation system ran out of memory, causing the test run to fail. Some of the students who were running the performance monitor of the Windows Task Manager noticed that the CPU utilization on their machines was 100% while JMeter was generating loads of more than 500 users in their simulation. Of course, the CPU utilization depends on the hardware configuration on the students' individual machines. This is a valuable lesson. It shows that it is necessary for the load generation host to have sufficient processing power and memory to run the tests at the desired loads.

One possible reason for run failures is that all students in attendance ran load tests simultaneously. This increased response times, and hence the number of uncompleted transactions at any instant. With five hundred virtual users per test, 45 students, and two test loops per student, this means that the tested website was receiving a load of at least 45,000 (i.e. 500* 2 * 45) requests per second. Some students reported similar results when testing little known or seldom visited websites in their homework. These sites were likely hosted on less powerful configurations than the one hosting the website that was tested in class.

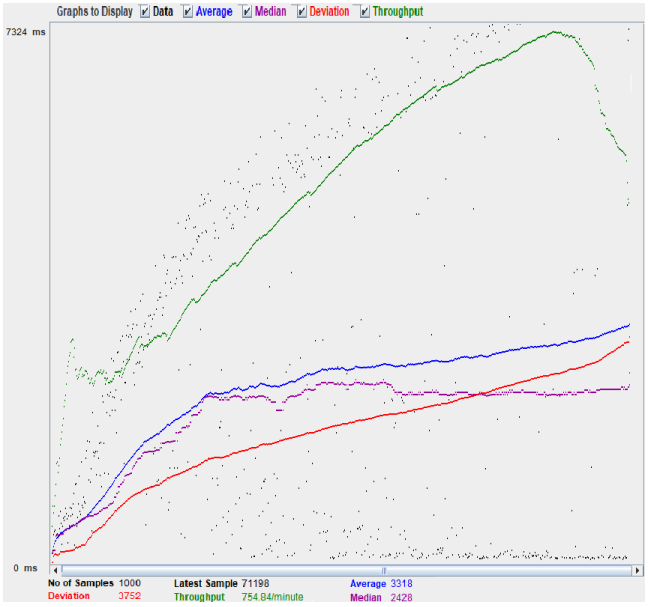

Figure 5: Result of performance testing under scenario 3, 500 users and 2 loops

The foregoing figures show that the rate of increase in outbound packet throughput, shown by the green curves, diminished over time, until the load was stopped. Then, the packet throughput gradually dropped off. By contrast, the response times increased almost monotonically. The outbound packet throughput is a surrogate quantity for the page hit rate, while the inbound packet rate is a surrogate quantity for the page return rate.

5 ETHICAL CONCERNS

Generating page requests to a website one does not own poses ethical questions of which students should be made aware.

(1) Students should not attempt to saturate any web site they do not own. Some well known websites such as LinkedIn have user agreements that prohibit users from placing an unreasonable load upon them [12], while others, such as Google [8], prohibit the automatic generation of user queries without express permission [17]. Thus, students should choose websites for "target practice" with care. Moreover, they should not generate transactions too rapidly. This could be construed as a denial of service (DOS) attack and potentially make them targets of a criminal investigation. Their IP addresses could be denied access to the service altogether [13]. The use of JMeter to generate a DOS attack is described in the open literature [9].
(2) Many websites contain objects that must be retrieved from a content delivery network (CDN). The site owner may have to pay for delivery every time a CDN-hosted object is sought [15].
(3) Even if the user owns the target website, the hosting terms of service may require that the site not be subjected to large volumes of test traffic.

We found it necessary to caution students about these issues after learning that one of them had crashed a website owned by a friend.

6 DISCUSSION AND RECOMMENDATIONS

Our motivation for having students measure resource usage while playing recordings of the same operatic music with video and with sound alone is that doing so is a convenient way for them to see that one has a very different resource usage pattern from the other. Testing websites with JMeter at various loads allows them to see that a surge in traffic can cause performance degradation as observed by the users. Unfortunately, we could not provide server-side measurements of the corresponding resource utilizations to motivate the use of basic performance modeling to plan performance tests. We could have partially overcome this issue by having the students measure the CPU and bandwidth utilizations as functions of the transaction generation rate or of the number of virtual users running on their own machines. That would have demonstrated that the load they could generate is limited by the compute power of the load generators as well as of the capabilities of the system under test. Also, we would have liked to combine the explanation of the design of performance testing with in-class exercises and a detailed exposition of asymptotic bottleneck analysis to motivate the design of performance tests. There was insufficient class time to do this. We recommend that at least six hours of class time be devoted to performance modeling in support of testing, and that this time should be reinforced by related exercises done at home.

When a course including performance testing exercises is taught at a university, concerns about targeting response time measurement experiments at a site one does not own could be overcome by choosing only university-owned pages as targets (with administrative permission obtained in advance), or by targeting sites that have been created for that purpose with the consent of their owners. The targeted pages could be available to the general public on request, on a site accessible only to the university community, or on an experimental site that is part of a software project in a lab. Alternatively, one could deploy an open source web site such as Sockshop

[16] or Train Ticket [7], on a university-owned host or on a virtual host provided by a commercial cloud provider [6]. Under no circumstances should server-side record updates or any other server-side modification be done as part of a student-generated load test. The instructors should work with the site owner's IT department to establish the suitability of particular pages for load testing and to provide traffic counts of page accesses. It would be instructive for students to know about the counts and the means of obtaining them so that they can understand the need to track offered traffic. If the system is in a lab with suitable resource measurement tools, it may be feasible to students to relate offered traffic to resource usage. In practice, doing so is essential to problem diagnosis and resolution.

Our personal contact with students in the Spring 2020 semester was limited by restrictions associated with the COVID19 pandemic. Therefore, we were only able to assess the students' interest in performance and how well they grasped it by looking at their measurement reports. At Stevens, it was clear that their experience of working with quantitative data was very limited for the most part. We are pleased to note that one student in the first author's section started measuring the behavior of applications that interested him. He also attended ICPE2020 on line and on his own initiative after asking his instructor how best to prepare to do so.

We note in passing that the terms used for the various parameters and outputs of JMeter and of other packages may differ from the terminology used in the standard performance literature. This complicates communication between those who have worked with one tool but not another. Concepts may have different names in different working environments. As a result, those who are focused principally on using particular tools rather than on underlying principles are at risk of making fundamental errors in the design and analysis of performance tests. Instructors should make an effort to reduce the risk of this occurring, especially as students are likely to encounter multiple technologies and suppliers of measurement instrumentation in the course of their careers. We also need to prepare students for the possibility that their efforts to base performance tests on underlying principles may encounter resistance in the workplace.

7 CONCLUSION

We have described our experience with including material on performance testing and measurement into a graduate level course on functional testing. We have shown students that performance tests should be based on a performance test plan driven by performance requirements, just as functional tests should be based on a functional test plan driven by functional requirements. We have given students a flavor of resource usage measurement, and shown them by example that adequate compute power and resources must be provided if load tests are to run to a successful conclusion. Moreover, our students have learned that there are restrictions and ethical considerations relating to the use of sites they do not own to practice performance testing techniques. The students have gained hands-on experience of comparing the resource usage incurred by two recordings with very different demand characteristics in a controlled manner, and of making conjectures about how they work. As instructors, we have learned that more class time and more in-home exercises are needed to give students a feel for how basic performance models can be used to plan performance tests, identify bottlenecks, and analyse the results. While we have not been able to provide our students a thorough grounding in performance modeling and testing, we have given them some experience of looking at performance data and taking measurements.

[6]We would like to thank the referees for this suggestion.

8 ACKNOWLEDGMENTS

The authors have benefited from discussions with Lu Xiao and Alex Podelko.

REFERENCES

[1] 2021 (accessed February 11t,h 2021). Performance Testing Market Size to Grow with Stupendous CAGR. (2021 (accessed February 11t,h 2021)). https://www.flanewsonline.com/performance-testing-market-size-to-grow-with-stupendous-cagr-key-driver-and-growth-forecasts/
[2] The President's Own United States Marine Band. 2009 (accessed December 16, 2020). *Grand March from Verdi's Aida (no video).* https://www.youtube.com/watch?v=TX0qN6QEvGg
[3] Len Bass, Paul Clements, and Rick Kazman. 1998. *Software Architecture in Practice.* Addison-Wesley.
[4] Francesco Adalberto Bianchi, Alessandro Margara, and Mauro Pezzè. 2017. A survey of recent trends in testing concurrent software systems. *IEEE Transactions on Software Engineering* 44, 8 (2017), 747–783.
[5] A. B. Bondi. 2014. *Foundations of Software and System Performance Engineering: Process, Performance Modeling, Requirements, Testing, Scalability, and Practice.* Addison-Wesley.
[6] P. J. Denning and J. P. Buzen. 1978. The Operational Analysis of Queueing Network Models. *Comput. Surveys* 10, 3 (1978), 335–261.
[7] FudanSELab. 2018. *Train TicketA Benchmark Microservice System.* https://github.com/FudanSELab/train-ticket
[8] Google. 2020 (accessed December 16, 2020). *Automated Queries.* https://support.google.com/webmasters/answer/66357?hl=en
[9] S. Grabovsky, P. Cika, V. Zeman, V. Clupek, M. Svehlak, and J. Klimes. 2018. Denial of Service Attack Generator in Apache JMeter. In *2018 10th International Congress on Ultra Modern Telecommunications and Control Systems and Workshops (ICUMT).* 1–4. https://doi.org/10.1109/ICUMT.2018.8631212
[10] Awni Hammouri, Mustafa Hammad, Mohammad Alnabhan, and Fatima Alsarayrah. 2018. Software bug prediction using machine learning approach. *International Journal of Advanced Computer Science and Applications* 9, 2 (2018), 78–83.
[11] Paul C Jorgensen. 2018. *Software testing: a craftsman's approach.* CRC Press.
[12] LinkedIn. 2020 (accessed December 16, 2020). *LinkedIn User Agreement.* https://www.linkedin.com/legal/user-agreement
[13] Olivia Morelli. 2020 (accessed December 16, 2020). *Your computer or network may be sending automated queries – how to fix?* https://ugetfix.com/ask/your-computer-or-network-may-be-sending-automated-queries-how-to-fix/
[14] Metropolitan Opera. 2014 (accessed December 16, 2020). *Verdi Opera Aida - Gloria all' Egitto, Triumphal March - HD (video).* https://www.youtube.com/watch?v=czEfHr8YGPA
[15] Alexander Podelko. 2020. Private Communication.
[16] Weaverworks. 2017. *Sock Shop: A Microservices Demo Application.* https://microservices-demo.github.io/
[17] Lu Xiao. 2020. Private Communication.