Software Engineering II - Lecture 10 Study Notes

Administrative Items

- Technical Seminar: Week of Oct 31
  - Purpose: Share knowledge and experience among teams.
  - Duration: 15 min presentation and 2 min Q&A.
  - Evaluation: usefulness, presentation quality (70% by peers, 30% by instructor).
  - Schedule: Technical Seminar Schedule
  - Submission: UMLearn -> Assignments -> Technical seminar.

Software Performance Engineering (SPE)

Introduction to SPE

- Importance of software performance over mere functional correctness.
- Example issues: slowed page load costing billions, Surface Pro battery-life issues.

Definitions

- **Performance Efficiency (ISO 25010)**: How effectively software utilizes resources under stated conditions.
- Performance indicators:
  - **Response Time**: Time taken from sending a request to receiving a response.
  - **Throughput**: Number of transactions processed per unit time (e.g. Transactions/Sec).
  - **Utilization**: Busy time ratio to total time.
  - Other: Capacity, battery/power consumption.
- Performance can be considered for the entire system or a specific component.

Objectives of SPE

- Increase revenue by timely processing.
- Avoid deployment and maintenance delays/costs due to performance issues.
- Early identification of bottlenecks.
- Improve server capacity and system reliability.

SPE Tasks Across SDLC

1. Non-functional Requirements Gathering
2. Design for High Performance
3. Unit Performance Test & Code Optimization
4. Performance Test
5. Monitoring & Capacity Management

SPE Phases

- **Requirements Phase**: Set performance goals, identify tools, plan resources and infrastructure.
- **Architecture and Design Phase**:

    - Assess performance implications of design choices.
    - Optimize algorithms, consider parallel processing, scalability, caching, load balancing.
- **Testing Phase**:
    - Create and simulate performance tests aligned with non-functional requirements.
    - Performance regression: ensure newer versions do not degrade performance.
- **Monitoring & Capacity Management**: Ongoing performance assessment and infrastructure planning for growth.

Testing Methodologies

- Types of tests: Stress, Endurance, Spike, Capacity, Configuration, Load Testing.
- Load Testing: Analyze system behavior under expected workload scenarios.
- Workload design considerations: Mix, intensity, step-wise or steady application of load.
- Identifying issues: Deadlocks, memory leaks.
- Performance modeling: Using machine learning to predict anomalies.

Performance Test Process

- Coordinated live-user and driver-based test execution.
- Use monitoring tools (e.g., JConsole, CA Willy, App Dynamics) and collect metrics/logs.

Measurements in SPE

- Static vs. Dynamic measurements.
- Internal vs. External measurements.
- Metrics: Response Time, Throughput, Resource Utilization etc.

Measurement Process and Bias

- Define the objective, tools, experimental variables, test cases, and analyze results.
- Address potential measurement bias and ensure reliable, reproducible results.

SPE Challenges

- Meeting time to market and budget constraints.
- Bridging the gap between developer feature knowledge and quality expert performance knowledge.
- Reducing manual validation with automation.

Resources

- Jiang ZM. Load testing large-scale software systems
- Jiang ZM, Hassan AE. A survey on load testing of large-scale software systems
- Additional resources linked throughout topics for detailed understanding.

Tips for Exam Success

- Understand the difference between various testing methodologies and their applications.
- Be familiar with performance metrics, how they are measured, and their significance.
- Be able to identify appropriate testing methods for a given scenario.
- Comprehend the importance of SPE throughout the software development lifecycle.

- Be capable of designing realistic loads based on historical data and estimating workload mix.

Class Notes

Slide 1

Title: Comp 4350 Software Engineering II Lecture 10
Dr. Shaowei Wang

Slide 2

Title: Administrative items
Technical seminar (the week of Oct 31)
The purpose of the seminar is to share knowledge and experience among teams. The topic of the seminar could be technology, design choice, challenges you faced and how you solved them, and lessons have learned, etc.
The seminar should be 15 mins (~13 minutes for presentation, 2 mins for Q&A) for each team.
The presentation will be evaluated in three aspects:

1. Usefulness of the seminar
2. Quality of the presentation
   Each team will be evaluated by other teams and the instructor (the weight is 70% : 30%), in terms of the usefulness of the seminar and presentation quality.
   Schedule could be found in the google sheet
   Submit your slides to UMLearn -> Assignments -> Technical seminar

Slide 3

Title: (Ultra) Large-Scale Software Systems
450 million active users

> 50 billion messages every day
> 4 million users
> 2600-3000 req/sec on most weekdays

Slide 4

Title: Software system failures are often due to performance issues rather than functional bugs
(Gp:) Loading

Slide 5

Title: A page load slowdown of only one second could cost?
$1.6 billion

Slide 6

Title: The short-battery-life of the Surface Pro tablet was due to a software performance bug

Slide 7

Title: Outline
Software Performance Engineering (SPE)
Performance testing
Load testing

Slide 8

Title: Software Performance Engineering (SPE)

Slide 9

Title: Performance Efficiency (ISO 25010

)

Slide 10

Title: Software performance
Performance measures the efficiency of the software against the constraints of time and resource allocation.
There are several indicators to capture and evaluate performance:

- Response Time: The total elapsed time between submission of a request and receipt of the response.
- Processing Rate/Throughput: The total completions per unit time, e.g. Transactions/Sec.
- Utilization: The ratio of busy time to total time (how busy or free the resources within a given system are).
  Other indicators (e.g., capacity, battery/power consumption)
  It is possible to consider the performance of:
- Entire system (including hardware and software)
- Part of the system such as a software component.

Slide 11

Title: Software Performance Engineering (SPE)
The set of tasks or activities that need to be performed across the Software Development Life Cycle (SDLC) to meet the documented Performance Requirements.

Slide 12

Title: SPE: Objectives

- Increase revenue by ensuring the system processes all transactions in a timely manner.
- Eliminate delayed deployment due to performance issues.
- Reduce the increased costs of maintenance due to performance issues during production or ad hoc performance corrections.
- Reduce operational overhead to address system problems due to performance issues.

- Identify bottlenecks by simulating a prototype.
- Increase server capacity.

Slide 13

Software Development Life Cycle
Performance Engineering Life Cycle
Functional Requirements Gathering
Architecture & Design
Implementation
System Test & User Acceptance Test
Deploy Into Production
Non-functional Requirements Gathering
Design for High Performance
Unit Performance Test & Code Optimization
Performance Test
Monitoring & Capacity Management
Source

Slide 14

Software Development Life Cycle
Performance Engineering Life Cycle
Functional Requirements Gathering
Architecture & Design
Implementation
System Test & User Acceptance Test
Deploy Into Production
Non-functional Requirements Gathering
Design for High Performance
Unit Performance Test & Code Optimization
Performance Test
Monitoring & Capacity Management

Slide 15

Title: SPE: Requirements Phase
Review production performance metrics of the current version if available.
Determine non-functional requirements so that system performance goals can be set and measured against.
Identify tools, resources, and infrastructure.
Early identification allows budget and time allocation for installation and staff training.

Slide 16

Software Development Life Cycle
Performance Engineering Life Cycle
Functional Requirements Gathering

Architecture & Design
Implementation
System Test & User Acceptance Test
Deploy Into Production
Non-functional Requirements Gathering
Design for High Performance
Unit Performance Test & Code Optimization
Performance Test
Monitoring & Capacity Management

Slide 17

Title: SPE: Architecture and Design

- Evaluate the alternatives.
- Provide input from a performance perspective to the architecture being recommended.
- Determine the capacity of the required infrastructure.
- Define performance targets for developers and across application components in case any bottleneck happens.

Slide 18

Title: Key considerations for designing software

- Optimize algorithms: Choose efficient algorithms and optimize them.
- Use parallel processing: Distribute processing across multiple cores or processors.
- Consider scalability: Design the architecture for scalability.
- Use caching and load balancing: Cache frequently accessed data and distribute the workload across multiple servers.

Slide 19

Software Development Life Cycle
Performance Engineering Life Cycle
Functional Requirements Gathering
Architecture & Design
Implementation
System Test & User Acceptance Test
Deploy Into Production
Non-functional Requirements Gathering
Design for High Performance
Unit Performance Test & Code Optimization
Performance Test
Monitoring & Capacity Management

Slide 20

Title: SPE: Testing
Create performance tests to simulate the workload model.

Use the tests to validate the non-functional requirements.
Identify application bottlenecks.
Validate the impact of code and configuration changes on application performance.
Identify performance regressions

Slide 21

Title: What is a performance regression?
Old version
New version
Does the new version have worse performance than the old version?

Slide 22

Mozilla takes performance regression seriously!

Slide 23

Software Development Life Cycle
Performance Engineering Life Cycle
Functional Requirements Gathering
Architecture & Design
Implementation
System Test & User Acceptance Test
Deploy Into Production
Non-functional Requirements Gathering
Design for High Performance
Unit Performance Test & Code Optimization
Performance Test
Monitoring & Capacity Management

Slide 24

Title: SPE: Monitoring & Capacity Management
Perform performance monitoring to continuously assess software performance and identify system capacity limits.
Perform capacity management to provide required infrastructure capacity for business workloads' growth.
Provide production workload data to support the development of the next version.

Slide 25

Title: How to form it as a pattern mining problem?
Transaction data
If a callstack pattern is observed from many trace streams that cost a large amount of CPU, it is suspicious to be a highly impactful performance bug

Slide 26

Title: Benefits of SPE
Defining a clear set of non-functional requirements ensures successful development.
Focus on system performance during all phases prevents late and costly changes.
Delivering a functional and performing system as required adds full value to the customer.

Slide 27

Title: Performance Testing

Slide 28

Title: Performance Testing
All the tests and methodologies to measure, verify and validate the performance of the system.
It is part of SPE.
Objectives include demonstrating system compliance with performance criteria and identifying components causing poor performance.

Slide 29

Title: Testing: Types of tests

- Stress testing: Tests the system's limits beyond max load.
- Endurance testing: Tests under expected load for a long time.
- Spike testing: Tests the system's reaction to sudden load changes.
- Capacity testing: Tests to find the maximum capacity.
- Configuration testing: Tests the effect of configuration changes.
- Load testing: Tests performance under expected load.

Slide 30

Title: Stress testing
Emphasizes on robustness, availability, and error handling under heavy load.

Slide 31

Title: Characteristics of stress testing
Analyzes system behavior after a failure.
Ensures system recovery and appropriate error messages under stress.
Metrics: Time to failure, error rates, behavior under failure, and recovery time.

Slide 32

Title: Testing: Types of tests

- Stress testing: Tests the limits of the system's capacity.
- Endurance testing: Tests the system under the expected load for a long time.
- Spike testing: Tests the reaction of the system by sudden load changes.
- Capacity testing: Tests the system's maximum capacity.
- Configuration testing: Tests the effect of configuration changes.
- Load testing: Tests performance under specific loads.

Slide 33

Title: Endurance testing
Observes if an application can withstand its expected processing load for an extended period.

Slide 34

Title: Endurance testing
Identifies performance problems during extended use.
Considers memory consumption to predict potential failures.

Slide 35

Title: Looking for known patterns: Deadlocks and memory leak
Performance data under steady load
[Avritzer et al., 2012]

Slide 36

Title: Deadlocks and memory leak: before and after fix
Before fix
After fix
[Avritzer et al., 2012]

Slide 37

Title: Testing: Types of tests

- Stress

testing: Tests the limits of the system's capacity.

- Endurance testing: Tests the system under the expected load for a long time.
- Spike testing: Tests the reaction of the system by sudden load changes.
- Capacity testing: Tests the system's maximum capacity.
- Configuration testing: Tests the effect of configuration changes.
- Load testing: Tests performance under specific loads.

Slide 38

Title: Spike testing
Analyzes system behavior when load is increased substantially and rapidly.

Slide 39

Title: Characteristics of spike testing
Assesses if the system can handle sudden load increases.
Analyzes system performance during and after the spike.
Metrics: Response time during the spike, system stability after the spike.

Slide 40

Title: Spike testing
Assesses the system's resilience to handle unexpected, rapid load changes.

Slide 41

Title: Testing: Types of tests

- Stress testing: Tests the limits of the system's capacity.
- Endurance testing: Tests the system under the expected load for a long time.
- Spike testing: Tests the reaction of the system by sudden load changes.
- Capacity testing: Tests the system's maximum capacity.
- Configuration testing: Tests the effect of configuration changes.
- Load testing: Tests performance under specific loads.

Slide 42

Title: Capacity testing
Evaluates the system's ability to handle increasing load volumes.

Slide 43

Title: Characteristics of capacity testing
Determines the system's breaking point. Identifies the maximum load the system can handle. Metrics: Response time at full capacity, error rates at full capacity.

Slide 44

Title: Looking for known patterns: Degradation by load increase
Performance data under increased load
[Avritzer et al., 2012]

Slide 45

Title: Testing: Types of tests

- Stress testing: Tests the limits of the system's capacity.
- Endurance testing: Tests the system under the expected load for a long time.
- Spike testing: Tests the reaction of the system by sudden load changes.
- Capacity testing: Tests the system's maximum capacity.
- Configuration testing: Tests the effect of configuration changes.
- Load testing: Tests performance under specific loads.

Slide 46

Title: Configuration testing
Analyzes performance variations due to system configuration changes.

Slide 47

Title: Characteristics of configuration testing
Evaluates performance changes due to configuration alterations. Ensures system stability after

configuration modifications. Metrics: Response time before and after configuration change, error rates after configuration change.

Slide 48

Title: Looking for known patterns: Performance variation by configuration change
Performance data before and after configuration change
[Avritzer et al., 2012]

Slide 49

Title: Testing: Types of tests

- Stress testing: Tests the limits of the system's capacity.
- Endurance testing: Tests the system under the expected load for a long time.
- Spike testing: Tests the reaction of the system by sudden load changes.
- Capacity testing: Tests the system's maximum capacity.
- Configuration testing: Tests the effect of configuration changes.
- Load testing: Tests performance under specific loads.

Slide 50

Title: Load testing
Evaluates system performance under expected conditions.

Slide 51

Title: Characteristics of load testing
Verifies if the system meets performance expectations. Ensures system functionality under expected loads. Metrics: Response time under expected load, error rates under expected load.

Slide 52

Title: Load testing
Validates that the system functions correctly under expected loads.

Slide 53

Title: Summary: Performance Testing Types

- Stress testing: Tests system limits.
- Endurance testing: Tests under expected load for an extended time.
- Spike testing: Tests sudden load changes.
- Capacity testing: Tests maximum capacity.
- Configuration testing: Tests impact of configuration changes.
- Load testing: Tests performance under expected loads.

Slide 54

Title: Tools for Performance Testing

- Apache JMeter: Open-source load testing tool.
- LoadRunner: Performance testing tool by Micro Focus.
- BlazeMeter: Load testing platform for web and mobile applications.
- Gatling: Open-source load and performance testing tool.
- WebLOAD: Load testing tool for internet applications.

Slide 55

Title: Common Mistakes in Performance Testing

- Insufficient or inaccurate test environment setup.
- Unrealistic load modeling.
- Neglecting to simulate real user scenarios.
- Inadequate monitoring and measurement.
- Lack of proper analysis of test results.

Slide 56

Title: Conclusion
Performance engineering is integral to delivering high-quality software. Testing various aspects of performance ensures the system meets requirements. Utilizing appropriate tools and methodologies is crucial for effective performance testing. Continuous monitoring and analysis help maintain optimal system performance.

Slide 57

Title: Thank You! Any Questions?