Lab 5

Performance engineering

Performance engineering

Performance: refers to the responsiveness of the system

Performance engineering within systems engineering, encompasses the set of roles, skills, activities, practices, tools, and deliverables applied at every phase of the systems development life cycle which ensures that a solution will be designed, implemented, and operationally supported to meet the non-functional performance requirements defined for the solution.

In other words Performance Engineering is the practice of applying Software Engineering principles to the product life cycle in order to assure the best performance for a product. The purpose is to know at each stage of development the performance attributes of the product being built.

Performance Engineering depends on a combination of verification and validation. For those of you who've forgotten this nuance, here's a brief review:

- Validation: Showing at project completion that the performance meets the stated goals.
- Verification: Showing at each stage in the development that the projected performance will meet the previously stated goals.

Performance engineering objectives

- Increase business revenue by ensuring the system can process transactions within the requisite timeframe
- Eliminate system failure requiring scrapping and writing off the system development effort due to performance objective failure
- Eliminate late system deployment due to performance [disambiguation needed] issues.
- Eliminate avoidable system rework due to performance issues
- Eliminate avoidable system tuning efforts
- Avoid additional and unnecessary hardware acquisition costs

- Reduce increased software maintenance costs due to performance problems in production
- Reduce increased software maintenance costs due to software impacted by ad hoc performance fixes
- Reduce additional operational overhead for handling system issues due to performance problems

Performance Engineering approaches and tasks

As performance engineering applies through the wholes software development life cycle (SDLC), we will go through methodology and approaches that covers what happened in each phase in SDLC. This methodology have multiple phases inception , elaboration, construction and transition these phases also known as Rational Unified Process[is an iterative software development process framework created by the Rational Software Corporation, a division of IBM since 2003]

1. Inception phase

This phase is considered as a conceptual phase that start by setting up the project scope, the constraints, the circumstance of the project and its environment and defining the critical business processes. High level risks that may impact system performance are identified and described at this time. Also define performance activities, roles, and deliverables. We perform performance analysis in this phase

Planning phases is the only phase attached to inception phase.

Planning phase: The start of any project is usually a good plan, so to make a successful IS without performance issues you should considered the following:

- Good IS contract management.
  - Define the scope clearly
  - Define time, budget and quality constraints.
- Good project management.
- Good process management:
  - Good selection of the appropriate technology.
  - Appropriate acquisition of technologies.

2. Elaboration Phase

During this defining phase, the critical business processes are decomposed to critical use cases. Such use cases will be decomposed further, as needed, to single page (screen) transitions. These are the use cases that will be subjected to script driven performance testing.

The type of requirements that relate to Performance Engineering are the non-functional requirements. While a functional requirement relates to what business operations are to be performed. A performance is related to nonfunctional requirement which indicate how fast that business operation performs under defined circumstances. Nonfunctional requirements are not limited to use cases. The overall system volumetric must be specified. These will describe the overall system load over a specified time period, defining how many of each type of business transaction will be executed per unit of time. The system volumetric documented in the Nonfunctional requirements documentation will be used as inputs for both load testing and stress testing of the system during the performance test.

To be able to perform performance test, we should start constructing a performance model using the use case as an input. The performance model is a simple black box. The desired output of the model includes throughput, response time, and availability.

In other words Performance modelling is a key technique to understanding problems in IT systems. Because it is difficult to estimate performance, IT systems must be designed with service levels in mind. In other words, a designer of an IT service must know the limits of the system a priori. For instance, a designer must know:

- The maximum number of transactions per second the system is capable of processing (i.e., an upper bound on throughput)
- The minimum response time that can be achieved by a transaction processing system (i.e., a lower bound on response time).

Some performance engineering activities related to performance testing should be executed in this phase. They include validating a performance test strategy, developing a performance test plan, determining the sizing of test data sets, developing a performance test data plan, and identifying performance test scenarios.

Requirement analysis and design are 2 phases that are included in elaboration phase.

- Requirement analysis phase: the goal of this phase is gathering the business requirements and be considered with the nonfunctional requirements also. There some questions that should answered at the end of this phase like :
    - Are Functional requirements identified?.
    - What are the performance requirements?
    - Do any functional requirements interfere with performance requirements?
    - What is the capacity planning guide for the system?
    - How much is a customer willing to pay for performance and scalability?

- Hardware
- Software licensing (e.g. OS, Oracle, etc.)
- System Administration

To be able answering this question we should:

- Involvement of users specifically in analysis phase.
- Good documentation of user requirement specification.
- Use the appropriate methodology, techniques, tools of systems analysis and conceptual design.

Also within the requirement gathering process, we have to set the performance goals that can be validated and verified later on. That's why these goals must be measurable and documented. To document this goals we perform performance inspections. Performance inspections are a technique for analyzing performance issues during the preparation of specifications. The goal of performance inspections is gather information needed to complete the performance documentation. There are guidelines to do performance inspections:

- These inspections should be conducted in a formal way within one meeting.
- There may well be questions generated that can only be answered by more thorough research.
- Experience shows an inspection requires several hours, with a few more hours to resolve action items.
- Be careful -- like any inspection, several people should be involved, including a dispassionate outsider.
- Be careful -- it's very possible to get so mired in details that the whole performance business becomes an overwhelming burden.
- Software developers have a way of being overly detail-conscious when it comes to gathering performance numbers.

- Design phase: we should choose the software architecture that will be used in this phase, to be able to build the performance model for it, and start constructing the performance test cases and seniors. Also produce the prototype that can be used for the first impression on how the system performance will be. Also the expected number of users or requests on system should be known or predicated to be able to measure performance. The main guidelines to go through this phase are :
  - The right selection of the technical specifications of system model.
  - The right documentation of the detailed design.

3. Construction Phase

In this phase, the main focus is on the development of components and other features of the system. This is the phase when the bulk of the coding takes place. In larger projects, several construction iterations may be developed in an effort to divide the use cases into manageable segments that produce demonstrable prototypes.

The construction and implementation phases are included within construction phase

- Construction phase: we start build the new software but never ignore or neglect to check the performance of your code in order to resolve any performance issue with the minimum cost. Performance testing and analysis must occur throughput construction

4. Transition

The primary objective is to 'transit' the system from development into production, making it available to and understood by the end user. The activities of this phase include training the end users and maintainers and beta testing the system to validate it against the end users' expectations. The product is also checked against the quality level set in the Inception phase.

Implementation and Maintenance phases are included in transition phase

- Implementation phase: install the environment and deploy the constructed code. Here also we should run performance test before Quality Assurance. In order to eliminate the discovered issues with QA to minimize the cost of resolving these issues.
  We should perform the performance testing in this phase where the system is in its beta version.
  There are some guidelines that applied in implementation phase:
    - Executing the development and testing of programs, through the effective technical programmers and testers.
    - Applying the configuration management requirements.
    - Applying the quality assurance measurements for software engineering.
    - Using the test cases approved by the user.
    - Applying all types of testing techniques during phases and activities of system development life cycle.
    - Applying the appropriate strategy of the installation, implementation, operation, training and delivery.
    - Closing the project according to the contract and start the maintenance phase if it's included in contract.

  After the installation and deployment activities is done, we should perform a testing for the system like UAT testing. So in testing activity the main goal is Identify and eliminate performance problems before they get into production. So a new case is s

    - Performance testing and analysis must occur throughput development!!!
    - In late cycle QA, should be a formality with no surprises.
    - A surprise at this point will delay product release or potentially kill a product.
- Maintenance phase: is about solving bugs and issues after the UAT and sign of the project. Which has different fees .the goal is to Identify and eliminate performance problems which detected by users.