1. **Difference between Load and Stress testing??**

**Answer**: The idea of stress testing is **to find the breaking point in order to find bugs** that will make that break potentially harmful. Load testing is merely testing at the highest transaction arrival rate in performance testing to see the resource contention, database locks etc.

1. **What's Load/Volume Test?**

**Answer**: Load/volume tests study how a program handles large amounts of data, excessive calculations, and excessive processing. These tests do not necessarily have to push or exceed upper functional limits. Load/volume tests can, and usually must, be automated.

Focus of Load/Volume Testing

* Pushing through large amounts of data with extreme processing demands.
* Requesting many processes simultaneously.
* Repeating tasks over a long period of time

Load/volume tests, which involve extreme conditions, are normally run after the execution of feature-level tests, which prove that a program functions correctly under normal conditions.

1. **What's Scalability and Performance Testing?**

**Answer**: Scalability and performance testing is the way to understand how the system will handle the load cause by many concurrent users. In a Web environment concurrent use is measured as simply the number of users making requests at the same time.

Performance testing is designed to measure how quickly the program completes a given task. The primary objective is to determine whether the processing speed is acceptable in all parts of the program. If explicit requirements specify program performance, then performance test are often performed as acceptance tests.

As a rule, performance tests are easy to automate. This makes sense above all when you want to make a performance comparison of different system conditions while using the user interface. The capture and automatic replay of user actions during testing eliminates variations in response times.

This type of test should be designed to verify response and execution time. Bottlenecks in a system are generally found during this stage of testing.

1. **Desktop application development and Test automation?**

**Answer**: The software was written to provide a friendly interface for information workers: Spreadsheet jockeys, business people needing written reports, and game players. The full spectrum of desktop software could pretty well be categorized into spreadsheet, word processor, database, and entertainment categories since desktop computers were rarely networked to other information resources. Desktop applications used the keyboard, and then later a mouse, to navigate through windows and a drop-down menu. Inside a desktop application software package one would find an event-driven framework surrounding individual procedural functions. The automation focused on improving the time it took to test a desktop application for functionality. The test utilities link into desktop applications and try each command as though a user were accessing the menu and window commands. Most QA technicians testing a desktop application compare the function of all the menus and windows to a written functional specification document. The variation from the document to the performance shows the relative health of a desktop application.

1. **When test a FoxPro database?**

**Answer**:

* If the database is linked to other database, are the links secure and working?
* If the database publishes to the Internet, is the data correct?
* When data is deployed, is it still accurate?
* Do the queries give accurate information to the reports?
* If the database performs calculations, are the calculations accurate?

1. **When testing an Access database?**

**Answer**:

* If the database is creating Web pages from the datbase to a URL, is the information correct and updated? If the pages are not dynamic or Active Server pages, they will not update automatically.
* If the tables in the database are linked to another database, make sure that all the links are active and giving reevant information.
* Are the fields such as zip code, phone numbers, dates, currency, and social security number formatted properly?
* If there are formulas in the database, do they work? How will they take care of updates if numbers change (for example, updating taxes)?
* Do the forms populate the correct tables?
* Is the database secure?

1. **When testing a SQL server?**

**Answer**:

* If the Web site publishes from inside the SQL Server straight to a Web page, is the data accurate and of the correct data type?
* If the SQL Server reads from a stored procedure to produce a Web page or if the stored procedure is changed, does the data on the page change?
* If you are using FrontPage or interDev is the data connection to your pages secure?
* Does the database have scheduled maintenance with a log so testers can set changes or errors?
* Can the tester check to see how back ups are being handled?
* Is the database secure?

1. **Recovery testing?**

**Answer**:

* The system recovers from faults and resumes processing within a predefined period of time.
* The system is fault-tolerant, which means that processing faults do not halt the overall functioning of the system.
* Data recovery and restart are correct in case of automatic recovery. If recovery requires human intervention, the mean time to repair the database is within predefined acceptable limits.

1. **Data validity?**

**Answer**: The most common data errors are due to incorrect data entry, called data validity errors

1. **What's Data integrity?**

**Answer**:

Data stored in the database should include such items as the catalog, pricing, shipping tables, tax tables, order database, and customer information. Testng must verify the integrity of the stored data. Testing should be done on a regular basis because data changes over time.

**Data integrity tests**

Data integrity can be tested as follows to ensure that the data is valid and not corrupt:

* Test the creation, modification, and deletion of data in tables as specified in the business requirement.
* Test to make sure that sets of radio buttons represent a fixed set of values. You should also check for NULL or EMPTY values.
* Test to make sure that data is save to the database and that each values gets saved fully. You should watch for the truncation of strings and that numeric values are not rounded off.
* Test to make sure that default values are stored and saved.
* Test the compatibility with old data. You should ensure that all updates do not affect the data you have on file in your database.

1. **How to Query responses time?**

**Answer**: The turnaround time for responding to queries in a database must be short; therefore, query response time is essential for online transactions. The results from this test will help to identify problems, such as possible bottlenecks in the network, specific queries, the database structure, or the hardware.

1. **Is Extreme Programming not for Software Quality Engineering and Software Configuration Management practitioners?**

**Answer**: XP is a development discipline that is for customers (in their role as specifiers and their role as investors and their role as testers and acceptors) and for developers. As such, the Quality Engineering and Configuration Management roles are critical to the effort. They have to be assigned and played in a way that is consistent with the mission of the group, the level of criticality of quality, and so on. We'd need to talk in detail about your situation to see just where the XP terminology connects with yours, but your QA functions need to be done in any effective software effort, whether in a separate organization or not. So XP certainly is for software quality engineering and software configuration management, as part of a healthy overall process. That said, XP is aimed at smaller projects (like yours) and it sounds like yours has a much higher level of QE and CM than is often seen in companies of your size. That should give you a strong leg up in building quality software, and we should strengthen your contribution to quality as we bring XP into the team.

1. **How can a person write storyboards without having the basics of pinpointing and developing sound requirements?**

**Answer**: Do you use IEEE, SEI, ISO9000 standards as references to acquire the fundamentals of defining accurate requirements for customers and software engineering users? How can a person write storyboards without having the basics of pinpointing and developing sound requirements?   
  
We would agree that those who play the customer role have to know what they want. We do not, however, recommend any particularly formal requirements writing or recording mechanism. Instead, what we are working toward (XP is for small teams, after all) is to have a clear understanding in the heads of customers, developers, and testers as to what is wanted.   
  
Rather than have, say, an "analyst" sit down with the customer and laboriously translate his mumblings into something representing what is wanted, and then having a "designer" take the analysis and build a design, and so on, small teams function best if the customers and designer/developers talk to one another until they develop a common vocabulary of what is needed and how it will be done. In XP, we would like to have a common level of understanding in all heads, each focused on its own particular interests:   
  
Customers: what's needed, what's the business value, when do we need it?  
Developers: what's needed, how can I build this, how can I test my code, how long will it take?  
Testers: what's needed by the customers, how can I test whether developers have done it?   
  
As you can see, the testers' functional tests are what close the loop, assuring everyone that what was asked for was what we got. The best way to do XP is with a separate functional testing organization that is closely integrated into the process. It would be delightful to have that organization run by an experienced QA manager trained in XP.

1. **What are the Quality Assurance and Software Configuration Management roles and responsibilities with Extreme Programming?**

**Answer**: We prefer for there to be a separate organization for functional testing (probably exactly like your QA function, with testing results made public very quickly). XP, however, only says that there must be functional tests: it does not specify organizationally how they must be done. Experience is that testing is best done by a separate function - but one that is very tightly integrated with development rather than at the end of a long pipeline.   
Configuration management is also up to the team. It is usually necessary to have one or more individuals responsible for CM. We have no special rules or practices addressing how a group would manage the requirement to build multiple systems from one code base. Our main approach would be: for each release configuration, there must be corresponding functional tests, and these must be run before that configuration is released to the (real) customer. We would think that development would proceed by running kind of a "union" of all the functional tests of all the configurations.   
We'd probably have to talk more specifically about how your specific organization needs to build configurations to say much more about that.

1. **When do the all the customer sign-offs occur?**

**Answer**: Customer sign-off is continuous. Each iteration has its functional tests. Everyone is fully up to date on which tests are working and which are not. If tests scores are trailing implementation by too much, the customer will inevitably schedule more work against older features that are incorrect (or whose requirements have changed). When test scores are tracking implementation, the customer knows it and is comfortable requesting new functionality.   
Because the test scores are public and visible, everyone has the same level of understanding of where quality is. Generally scores are showing a good curve toward release, and everyone gets increasing comfort as the release date shows up. And, of course, if tests are not tracking, everyone knows that and the priority of getting things right naturally increases.   
The overall idea of this part of the process is to provide the most rapid feedback possible to everyone, customers and developers alike. That's why we like all the functional test run every night. Next morning, if anything has been broken the day before, everyone knows it and can deal with it effectively (since it was only yesterday's work that could be the problem). The faster the feedback, the faster development of quality software can proceed.

1. **Within XP, are there any specification baselines, test baselines, QA Acceptance testing, and CM Release Management/Change Control?**

**Answer**: XP is an inherently incremental process, with software being released to "production" as frequently as possible. This generally means that programmers release their work to the common development pool approximately daily, and that means that if a full system were built on that day, their code would be included in that build. The time period between full system builds varies depending on the environment: since you have chosen a particularly difficult integration language (C++), I could imagine that you would build less frequently. We would recommend, however, that the full system be integrated as often as possible, at least daily. (This may seem aggressive to you. We'd have to talk about what is possible in your environment.)   
Since XP is incremental, developers are working in short time increments we call iterations: we recommend about three weeks. Features (user stories) are broken down to the point of detail that allows a developer and his partner to implement the stories they're working on in that time period. We like the functional tests for that iteration to be complete and available no more than half-way through the iteration. (This usually means that QA is writing tests for the next iteration while this one is going on.)   
All through the iteration, programmers can use QA's functional tests to determine whether they have met the requirements. (They are also using their own unit tests to determine whether their individual classes are doing what they should. This is usually at a much finer level of detail.)   
Baselines work this way: when the code for a story is released, all the functional tests for it should be in place, and will ideally be working. Inevitably some will not, especially with teams just beginning with XP. One of the quality measures in the process is the daily graph of performance on functional tests. The general shape of this graph, over the course of the full system release period, is that of two s-curves: the upper curve is the total number of tests written, the lower curve is the number running at 100%. A healthy project of course shows these curves coming together at 100% by the end of the schedule.   
The code management software needs of course to reflect the requirements scheduled for release. This is determined by the "customers", as part of the planning components we call the commitment schedule (overall plan for a major release) and the iteration plan (plan for a (three week) iteration). The baseline of what is in the system tracks what is actually requested by the customers. Development doesn't care whether this is new functionality or a change to old. They don't care whether a given user story addresses something that was planned for or not. XP is completely flexible with regard to change management: development merely estimates how long any desired feature will take, and works on it when "customer" schedules it into iteration. (Dependencies of course exist, but we find that far fewer exist than most developers believe. Drilling into that subject is beyond the scope of this email.)

1. **How is Software Quality Assurance and Software Configuration Management integrated into Extreme Programming?**

**Answer**: XP defines two levels of testing. The first is unit testing, which must be performed by the programmers as they work. Each class implemented must have programmer-developed unit tests, for everything that "could possibly break". These tests are to be written during coding of the class, preferably right before implementing a given feature. Tests are run as frequently as possible during development, and all unit tests in the entire system must be running at 100% before any developer releases his code.(By release, we mean transferring from his own code space to the code integration area. This is handled differently, of course, depending on the code management tools in place.) The second level of testing is called functional testing. Each feature of the system (which is defined by something we call a User Story, rather like a Use Case) must have one or more functional tests that test it. The functional tests are the responsibility of what we call the "customer", the body responsible for defining the requirements. The implementation and running of functional tests can be done by the Software QA group, and in fact this is an ideal way to do it.

1. **What's System-Level Test?**

**Answer**: A System-level test consists of batteris of tests that are designed to fully exercise a program as a whole and check that all elements of the integrated system function properly.

**Functional System Testing**

System tests check that the software functions properly from end-to-end. The components of the system include: A database, Web-enable application software modules, Web servers, Web-enabled application frameworks deploy Web browser software, TCP/IP networking routers, media servers to stream audio and video, and messaging services for email.

A common mistake of test professionals is to believe that they are conducting system tests while they are actually testing a single component of the system. For example, checking that the Web server returns a page is not a system test if the page contains only a static HTML page.

System testing is the process of testing an integrated hardware and software system to verify that the system meets its specified requirements. It verifies proper execution of the entire set of application components including interfaces to other applications. Project teams of developers and test analysts are responsible for ensuring that this level of testing is performed.

System testing checklist include question about:

* Functional completeness of the system or the add-on module
* Runtime behavior on various operating system or different hardware configurantions.
* Installability and configurability on various systems
* Capacity limitation (maximum file size, number of records, maximum number of concurrent users, etc.)
* Behavior in response to problems in the programming environment (system crash, unavailable network, full hard-disk, printer not ready)
* Protection against unauthorized access to data and programs.

1. **Ping tests?**

**Answer**: Ping tests use the Internet Control Message Protocol(ICMP) to send a ping request to a server. If the ping returns, the server is assumed to be alive and well. The downside is that usually a Web server will continue to return ping requests even when the Web-enable application has crashed.

1. **What's Web-Enabled Application Measurement Tests?**

**Answer**:

1. Meantime between failures in seconds
2. Amount of time in seconds for each user session, sometimes know as transaction
3. Application availability and peak usage periods.
4. Which media elements are most used (for example, HTML vs. Flash, JavaScript vs. HTML forms, Real vs. Windows Media Player vs. QuickTime)