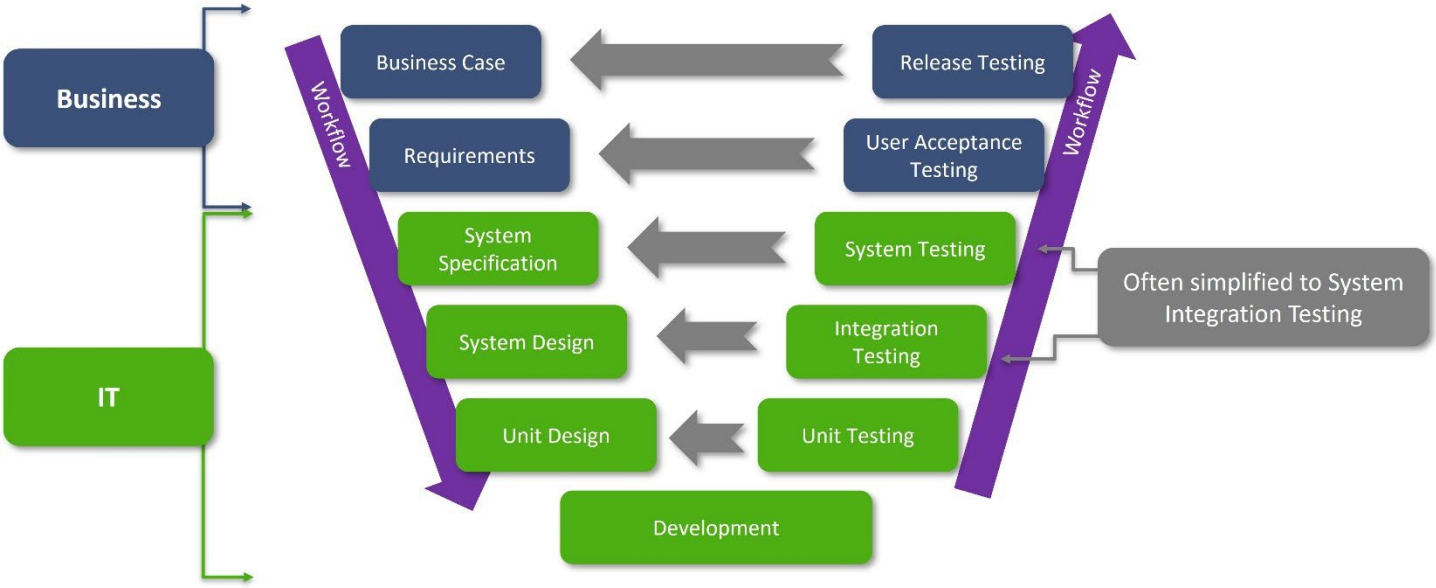
TREXIN INSIGHT PAPER

April 12, 2022

USER ACCEPTANCE TESTING VS. SYSTEM INTEGRATION TESTING

Exploring the differences between two prominent components of any software development project and why they shouldn't be performed simultaneously.

Technology and business have evolved to be evermore interdependent of one another and the importance of understanding the different components and roles/responsibilities within the software development lifecycle have increasingly become paramount to ensure success and that all involved parties can deliver on expectations. User Acceptance Testing (UAT) and System Integration Testing (SIT) are two great examples to dive in deeper and review. UAT represents testing whether an application meets the business need whereas SIT represents the testing of an application to ensure it meets its engineering specifications. Both, while reliant on one another, are very different activities with different outcomes and must rely on different expertise to complete their individual tasks. To aide in our understanding of these components, it is helpful to employ a version of the V Software Development Diagram first created by the Hughes Aircraft company. Please see figure 1 below:



This figure illustrates a particular software development lifecycle; the boxes represent different phases, the purple arrows represent the direction of the workflow, the gray arrows represent which testing phase lines up to which phase of design/build, and the blue areas represent where the business side is in charge while the green represents the areas typically owned by IT.

SIT EXPLAINED

To further explain, SIT is commonly accepted as a combination of two different, but similar enough software testing phases, Integration Testing and System Testing. During SIT, the primary focus is on making sure that the different components of the application work with each other effectively and match their technical design specs. This stage of testing should ask and answer questions such as: "does server d751 run widget software 2 to transform document d into

a PDF and transmit it to server a123 with no data loss?" or "is this connection between the database and the user desktop https encrypted and sending accurate data?". The goal of SIT is to achieve a 'pass' to all technical questions proving the software fulfills its technical design specs and the different system components function together properly. This fulfillment is usually measured by a group of tests that must be performed on the software and many companies will have a formalized, or even automated process for completion, approval, and hand-off to UAT. One notable area of difference between approaches is that in an Agile development environment SIT is typically performed every sprint rather than one-time efforts near the end of the waterfall-based project, although there may be another, more thorough round of SIT towards the end of a project.

UAT EXPLAINED

After hand-off, UAT is focused on whether the application meets the broader needs of the business it was designed for (i.e. are the business requirements met). UAT testing should ask and answer questions such as "is the software accurately and reliably measuring whether we are meeting our reserve requirements and updating it every 5 minutes so that compliance can see it?" or "does the software enable me to share Word documents with team members working from home?". At the end of UAT, the end-users should be satisfied with the software; established through multiple pre-arranged tests where a representative sample of users use various aspects of the software within a testing environment. The formal documented approval process usually consists of a decision as to whether the software is "good to go" or not based on the results of the tests. If the software is deemed "good to go", then it's handed off to release testing and management where training will take place, and the software will begin being fully used by end-users. It should be noted that within an Agile development environment a company may perform UAT within every sprint, although this is often not the case due to budget constraints.

BRINGING IT TOGETHER

The examples above, whether an application is successfully connecting two servers, or whether it is providing adequate data to the compliance department to meet business specifications, are two very different questions and should not be solved within the same breath even though they both belong as essential pieces of a larger release. Not only should they be performed by different groups, but if case issues arise and troubleshooting needs to occur, having the appropriate separation of duties allows for faster resolution and work re-assignment/future hand-offs. The goal is to eliminate as much confusion as possible, not increase it. And to drive the point home, let's examine a non-software example:

> The Ancient Roman Senate has decreed that a stone bridge be constructed over the Tiber so that grain can be more easily transported, thus the bridge must be able to support the weight of 10 oxen upon it. The Senate then places an ambitious Equite named Titus as project manager for the bridge construction project, who promptly begins working on planning and building the bridge. Titus already has the business case and requirements from his stakeholders, the Senate, so he promptly begins working on the technical specifications alongside his development team; they determine that the bridge should be constructed of concrete blocks and several arched spans.
>
> After determining the technical specifications development begins and workers start building the bridge. Once the work is completed, it will be time for the bridge to be tested. The testing has four parts, the first is unit testing, which was actually completed during development. The individual concrete blocks (components) were tested to make sure they were one solid piece and didn't have any manufacturing defects. The second test is SIT where the arches of the bridge must be visually inspected to make sure all the blocks are properly in place and

none are missing or improperly joined. The third test is UAT, and the Senate will watch as 10 oxen are marched across the bridge at the same time to prove it can hold their weight. The fourth test is release testing, where Titus will see whether the bridge is capable of being used to supply grain to the city of Rome.

Titus, being ambitious, decides that he will save time by combining SIT and UAT, the bridge arches will be inspected for missing/broken components while the oxen march on top. Sadly, the bridge has a flaw, and as the oxen start walking on it, it collapses dramatically in front of the Senate's judging gaze. Because of the chaos no one can tell where the flaw was or what happened, and as the development team hadn't finished inspecting the arches before the oxen started walking, they don't know whether it was due to a missing component, improper joining, or due to the bridge having been improperly designed for the weight from the get-go. This is a serious setback for the project, so it is cancelled due to fears of cost overflow, and Titus is fired.

Even if the above story sounds unconnected to modern software development, the goal is to showcase that performing key testing activities at the same time is difficult for many of the same reasons that Titus couldn't check whether the bridge had been assembled correctly and whether it could hold weight simultaneously. Trying to evaluate two distinct steps requires non-technical business users to become technical and determine whether the software works before its even known that the software does what the developers intended. Don't add to confusion and frustration and assign roles and tasks appropriately to avoid low buy-in, poor engagement, and inevitable project delays. Even though at first glance it may look like it will cost more and take longer, testing in the proper order will ultimately make any software release run smoother, more efficiently and deliver a stronger end product.

Interested in learning more about software testing, or expanding your organization's technological enablement? Trexin can help. Contact us today.


This TIP was written by Jay DeLuna and Duncan Goeden. Jay and Duncan welcome comments and discussion on this topic and can be reached at jay.deluna@trexin.com and duncan.goeden@trexin.com.