Regresyon Testi: Yazılımda herhangi ufak bir değişiklik ya da yeni bir fonksiyon birçok beklenmedik sonuç doğurabilir. Regresyon testinin amacı, bu değişiklikler sonrası yazılımın hala doğru şekilde çalıştığını kontrol etmektir. Regresyon testi canlıda çalışan kodun üzerinde yapılan değişikliklerin kontrolü için kullanılır. Bu değişiklikler yeni bir fonksiyon, hata çözümü ya da performans geliştirmesi olabilir. Regresyon testleri genellikle değişiklikler son aşamaya geldiğinde ve yazılımın yeni sürümü yayınlamadan önce gerçekleştirilir. Regresyon testlerinin öncelikli amacı, uygulamanın kritik alanlarının hala beklendiği gibi çalıştığını kontrol etmektedir. Bu testlerin maliyeti yüksek olduğu için spesifik bir alan seçilerek o alanda testler yoğunlaştırılır. Bunlar sık kullanılan alanlar, hata çıkması muhtemel uygulama alanları, ana ve karmaşık fonksiyonlar olabilir.

A/B Testi: Bir A / B testinde, bir web sayfası veya uygulama ekranı alıp aynı sayfanın ikinci bir sürümünü oluşturmak için onu değiştirilir. Bu değişiklik, tek bir başlık veya düğme kadar basit olabilir veya sayfayı tamamen yeniden tasarlayabilir. Ardından, trafiğin yarısı sayfanın orijinal sürümünde gösterilir (kontrol olarak bilinir) ve yarısı sayfanın değiştirilmiş halini gösterir. Ziyaretçilere kontrol veya varyasyon sunulduğunda, her bir deneyimle olan etkileşimleri ölçülür ve bir analiz tahtasında toplanır ve istatistiksel bir motorla analiz edilir. Daha sonra, deneyim değiştirmenin ziyaretçi davranışı üzerinde olumlu, olumsuz veya hiçbir etkisi olup olmadığı belirlenebilir.

Black Box/White Box Testleri: İkisi de Penetrasyon testidir. Peneterasyon testleri geliştirilen uygulamanın saldırılara ne kadar dayanıklı olduğunu, güvenlik açıklarını tespit etmek için uygulanır. Kara Kutu (Black Box) Yaklaşımı: Bu yaklaşımda sızma testinin yapılacağı sistem ile ilgili bir bilgi önceden verilmez. Test edici bir bilgisayar korsanı gibi sisteme sızmaya çalışır. Hedef sisteme sızmak için sistemle ilgili bilgi toplanır; zafiyetler ve açıklar taranır. Harici veya son kullanıcı bakış açısından test etmeyi içerir. Test edicinin yanlışlıkla sisteme zarar verme ihtimali bulunmaktadır. Beyaz Kutu (White Box) Yaklaşımı: Sızma testini yapacak ekibe firmada bulunan tüm sistemler hakkında bilgiler verilir. Black box yaklaşımına göre kuruma daha fayda sağlanır. Zafiyetleri bulmak kolaylaşır ve önlem alınması için geçen süre de azalmaktadır. Ekibin zarar verme riski çok azdır.

Mutasyon Testi: Kaynak koddaki belirli ifadeleri değiştirdiğimiz(mutant) ve test senaryolarının hataları bulabildiğini kontrol ettiğimiz bir tür yazılım testidir. Temelde birim testi (unit test) için kullanılan bir beyaz kutu testi türüdür. Mutant programdaki değişiklikler son derece küçük tutulur, bu nedenle programın genel hedefini etkilemez. Mutasyon testinin amacı, mutant kodunu kaldıracak kadar sağlam olması gereken test vakalarının kalitesini değerlendirmektir. Bu yöntem, programda bir hata oluşturmayı içerdiği için hata tabanlı test stratejisi olarak da adlandırılır.

Behavior Driven Development: BDD yaklaşımı ilk olarak 2009 yılında Dan North tarafından ortaya atılmıştır. Açılımı Behavior Driven Development olan BDD, TDD yaklaşımının karmaşıklığı gidermek amacıyla ortaya çıkmıştır. Yazılım süreçlerinin daha test odaklı gitmesini sağlayan bir yaklaşımdır. TDD yaklaşımında olduğu gibi burada da yazılım geliştirmeye başlamadan önce test senaryolarının yazılması desteklenmektedir. BDD'nin en güzel yanlarından biri konuşma dilinde test senaryoları yazmamıza olanak sağlamasıdır. İş analistleri müşteri ile yaptığı görüşmeler sonrasında ihtiyacı anlayarak user storyler oluşturmakta ve oluşturulan bu user storyler üzerinden de test senaryoları hazırlanarak koda dökülmektedir. Kısacası müşterinin ihtiyacı konuşma dilinde koda döküldüğü için müşteri ile ortak bir dilde buluşmaya olanak sağlayan bir yaklaşımdır aslında. Kolay bir yazım yöntemi olmasıyla birlikte "Given", "When", "Then" olarak 3 başlıkta senaryolar kurulmaktadır.

Agile Test Quadrant: Agile ve Agile Testing ile birlikte yazılım testi sadece testerın sorumluluğu olmaktan çıkar kalite artık tüm takımın sorumluluğudur. Hata bulmaktan çok hatayı önlemeye çalışır böylece test süreçleri daha proaktif ve sistematik hale gelir. Agile prensiplerini referans alır. Müşteri ihitiyaçları ve önceliklerine odaklanmıştır. Test, projenin başlangıcında başlar ve test ile geliştirme arasında sürekli bir entegrasyon vardır. Waterfall test yöntemine göre sürekli bir döngü içinde olduğundan developer ve tester işbirliği içinde olur. Çevik Test Çeyrekleri, ekiplerin ihtiyaç duyulan testi belirlemesine, planlamasına ve uygulamasına yardımcı olmak için yararlı bir sınıflandırma sağlar.

Çeyrek Q1 − Birim Düzeyi, Teknolojiye Yönelik ve geliştiricileri destekler. Birim testleri bu Çeyreğe aittir. Bu testler Otomatik testler olabilir.

Çeyrek Q2 – Sistem düzeyi, işle ilgili ve uygun ürün davranışı. Fonksiyonel testler bu çeyreğe aittir. Bu testler manuel veya otomatiktir.

Çeyrek Q3 − Sistem veya Kullanıcı Kabul Düzeyi, İşe Yönelik ve gerçek zamanlı senaryolara odaklanma. Kullanıcı Kabul Testleri bu çeyreğe aittir. Bu testler manueldir.

Çeyrek Q4 − Sistem veya Operasyonel Kabul Düzeyi, Teknolojiyle Karşılaşılan ve Performansa Odaklanma, Yük, Stres, Sürdürülebilirlik, Ölçeklenebilirlik Testleri. Otomasyon testleri ile birlikte bu testler için özel araçlar kullanılabilir.