Journal of Information Systems and Management Research
Bilişim Sistemleri ve Yönetim Araştırmaları Dergisi

http://dergipark.gov.tr/jismar

Derleme Makalesi / Review Article

Kuantum Yazılım Test Teknikleri Ve Araçları Üzerine Bir İnceleme

Fatma Betül ÖZDEMİR[a*], Savaş ÖZTÜRK[b]

[a]*Marmara Üniversitesi, Teknoloji Fakültesi, Bilgisayar Mühendisliği Bölümü, İSTANBUL, 34854, TÜRKİYE*
bMarmara Üniversitesi, Teknoloji Fakültesi, Elektrik-Elektronik Mühendisliği Bölümü, İSTANBUL, 34854, TÜRKİYE

MAKALE BİLGİSİ

Alınma: 10.02.2025
Kabul: 27.04.2025

Anahtar Kelimeler:
Kuantum Programı, Kuantum Yazılım Testi, Kuantum Yazılım Mühendisliği

Sorumlu Yazar
e-posta:
fbetul23@marun.edu.tr

ÖZET

Kuantum bilgisayarlarının gelişimiyle birlikte kuantum yazılımlarının üretilme ve test ihtiyacı doğmuştur. Geleneksel yazılım test yöntemleri, kuantum mekaniğinin doğasından kaynaklanan karmaşıklık nedeniyle kuantum yazılımlarını test etmede yetersiz kalmaktadır. Kuantum yazılım sistemlerinin doğruluk ve güvenilirliğini sağlamak için kuantum test teknikleri ve araçları, özel olarak geliştirilmiş çözümler sunmaktadır. Bu çalışmada kuantum bilgisayar teknolojisinin geleceği için temel niteliği taşıyan güncel çalışmalar anlatılmıştır. Başlıca QuanFuzz, QMutPy, Quito vb. araçlar ile metamorfik, kombinasyonel ve arama tabanlı test tekniklerinden bahsedilmiştir. Kuantum test mühendisliği alanında görülen standartlaşma problemi üzerinde değerlendirmeler yapılmıştır. Bu çalışma, kuantum test teknikleri ve araçları üzerinde çalışma yapmak isteyen araştırmacılara bir rehber niteliğindedir.

DOI: 10.59940/jismar.1636890

A Review On Quantum Software Testing Techniques And Tools

ARTICLE INFO

Received: 10.02.2025
Accepted: 27.04.2025

Keywords:
Quantum Program, Quantum Software Testing, Quantum Software Engineering

Corresponding Authors
e-mail:
fbetul23@marun.edu.tr

ABSTRACT

With the development of quantum computers, the need to produce and test quantum software has arisen. Traditional software testing methods are inadequate for testing quantum software due to the inherent complexity of quantum mechanics. Quantum testing techniques and tools provide specially developed solutions to ensure the accuracy and reliability of quantum software systems. In this study, current works that are fundamental for the future of quantum computing technology are described. The main tools such as QuanFuzz, QMutPy, Quito, etc. and metamorphic, combinatorial and search-based testing techniques are mentioned. The problem of standardisation in the field of quantum test engineering is evaluated. This study is a guide for researchers who want to work on quantum testing techniques and tools.

DOI: 10.59940/jismar.1636890

GİRİŞ *(INTRODUCTION)*

Yazılım testleri, yazılım ürünlerinin belirlenen gereksinimleri karşılayıp karşılamadığını belirleyen, ürünün amaca uygunluğunu tespit eden, kusurlarını bulan, hem statik hem de dinamik süreç aktivitelerini kapsar.

Gelişen teknoloji ile çeşitli donanımların yerini artık yazılımlar almaya başlamış, mobil teknolojiler

günlük hayatımızın vazgeçilmez parçası haline gelmiştir. Örneğin, elektrikli otomobillerde, mekanik unsurlar yazılımla kontrol edilmeye başlanmış, bu durum geliştirici ve testçileri yazılımlarda oluşacak hataların can kaybına yol açacak kazalara karşı savunmasız kalma riski ile karşı karşıya getirmiştir. Yazılımsal başarısızlıklar can, mal ve itibar kayıplarına yol açarlar. Yazılımların test edilmesi ve kalitelerinin artırılması her zamankinden daha fazla önem kazanmıştır [1]. Geleneksel hesaplamadan önemli farklılıklar içeren kuantum yazılımlar, kendine has yöntemlerle test edilmelidir.

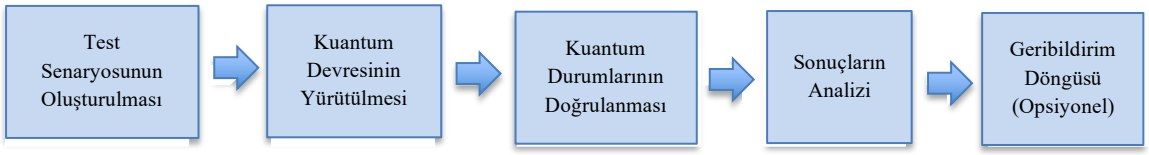
Son yıllarda yapay zeka ve makine öğrenmesi alanındaki önemli gelişmeler işlem gücü ihtiyacını da artırmış, daha hızlı bilgisayarlara ihtiyaç her zamankinden fazla olmuştur. 2025 yılının henüz başında ortaya çıkan DeepSeek adlı büyük dil modeli, pekiştirmeli öğrenme tabanlı öğrenme modeli ile rakiplerine işlem gücünü az gerektirmesi avantajıyla üstünlük kurmaya çalışmıştır. Diğer yandan OpenAI ve Meta gibi rakipleri de Nvidia gibi çip üreticilerinin satış stratejisini yönlendirerek pazar payını koruma stratejisi gütmektedir. Hesaplama gücü gereksinimleri, daha hızlı ve özgün alternatiflere ve özellikle de kuantum bilgisayarlara olan ilgiyi artırmıştır. Klasik yazılım testlerinde hata ayıklama deterministik bir süreçtir; bir girdinin çıktısı her zaman aynıdır. Ancak kuantum yazılımları, kuantum mekaniğinin olasılıksal doğasına sahip olduğu için, aynı girdiye karşılık gelen çıktı her zaman değişebilir. Bu durum, geleneksel test metodolojilerinin kuantum sistemlerine uygulanmasını zorlaştırır. Örneğin, bir klasik program hata içermediğinde her zaman aynı çıktıyı verirken, bir kuantum programı hata içermese bile ölçüm varyansları nedeniyle farklı çıktılar üretebilir. Bu nedenle, kuantum yazılım testi de klasik yazılım testinden ayrılır.

Kuantum Bilgisayarları (Quantum Computer - QC)[4], süperpozisyon (üst üste binme) ve dolanıklık (Entaglement)[2] gibi kendine has özellikleri olan çok sayıda karmaşık sorunu çözmeye çalışır. Örneğin; süperpozisyon; dinamik ya da statik bir sistemde, farklı doğrultu ve şiddetlerde etkilemekte olan birden fazla kuvvetin sistem üzerindeki etkilerinin ayrı ayrı belirlenip sonuçlarının toplamının alınmasıdır. Örneğin iki teknenin oluşturduğu dalgaların birbirinin içinden geçmesi ve birbirini etkilemesi oldukça karmaşık fizik hesaplarıyla çözülebilmektedir. Kuantum bilgisayarı, küçük ölçeklerde, fiziksel madde hem parçacıkların hem de dalgaların özelliklerini sergiler ve kuantum hesaplama, bu davranışı özel donanım kullanarak güçlendirir. Klasik fizik, bu kuantum aygıtlarının işleyişini açıklayamaz ve ölçeklenebilir bir kuantum bilgisayarı bazı hesaplamaları herhangi bir modern "klasik" bilgisayardan çok daha hızlı gerçekleştirme becerisini kuantum mekaniksel olgulardan alır[3]. Kuantum biti veya kısaca kübit, kuantum hesaplamasının temelidir. Klasik bir bit 0 ve 1'lerden oluşurken, kübitler ise $|0\rangle$ ve $|1\rangle$ biçiminde durumlardan oluşur. Bir kübitin genel durumu $|0\rangle + b|1\rangle$'dir. Karmaşık sayılar ile ifade edilen eşitlik; $|a|^2 + |b|^2 = 1$'i sağlayan genlik ifadesidir. Genlikler her baz durumunun olasılık oranını temsil eder. Kuantum aygıtlarında, kübitlerin bilgisi yalnızca ölçümle elde edilebilir. Bir kuantum sistemini ölçmek, karşılık gelen genliğe sahip olma olasılığı olan klasik değerlerin elde edilmesi ile mümkündür. Elde edilen durumlar, iki kübit dolanık olduğunda, tüm durum iki alakasız kübit olarak ayrılamaz [2]. Basitçe, iki nesnenin zaman ve uzaydan bağımsız olarak birbirinden haberdar olması, birbirini etkilemesi (kelebek etkisi) olarak ifade edilebilir. Birçok yazılım mühendisi kuantum devrelerini programlamak için gerekli bilgi ve tecrübeye sahip olmadığı için kuantum programlarının kalite güvencesine yönelik sistematik bir süreç henüz ortaya konmamıştır[4]. Bu amaçla, Kuantum Yazılım Mühendisliği (Quantum Software Engineering - QSE)[4], yazılım mühendislerinin kuantum yazılımları üretebilmeleri için daha yüksek bir soyutlama düzeyinde uygun yöntem ve araçlar sağlamayı amaçlamaktadır [4]. Kuantum algoritmalarındaki gelişmelere ve kaynak gereksinimlerinin optimizasyonuna rağmen, algoritmaların birçoğunun donanım gereksinimleri yakın vadeli kuantum bilgisayarlarının yeteneklerinin çok ötesindedir. hem kuantum hem de klasik kaynakları kullanarak geleneksel klasik bilgisayarların erişemediği özdeğer ve optimizasyon problemlerine varyasyonel çözümler bulmak için tasarlanmış hibrit kuantum-klasik algoritma olan varyasyonel kuantum özçözücü (Variational Quantum Eigensolver - VQE)[5]'ler geliştirilmiştir. VQE, erken kuantum bilgisayarlarının performansını keşfetmek ve algoritmaların belirli bir mimarinin güçlü yönlerinden yararlanmak üzere tasarlanmıştır[5].

Kuantum bilgisayarların, yazılımlarının ve testlerinin endüstrideki uygulamaları son kullanıcı düzeyinde karşılık bulmaya başlamıştır. IBM, kuantum yazılımlarını test etmek için IBM Quantum Experience platformunda kendi hata düzeltme ve test araçlarını geliştirmiştir. Örneğin, IBM Qiskit framework'ü, kuantum programlarının doğruluğunu test etmek için gürültüye duyarlı (noise-aware) test teknikleri kullanmaktadır[3]. Bu teknik, kuantum gürültüsünü minimize etmeye yönelik hata modellemeleri içermektedir. Buna karşılık, Google

Quantum AI ekibi, Sycamore kuantum bilgisayarı üzerinde kuantum hata oranlarını tespit etmek için metamorfik test yöntemlerini kullanmaktadır.

Kuantum yazılım test teknikleri ve araçları; yazılım geliştirmenin tüm aşamalarında (modelleme, analiz, test etme, hata ayıklama) kuantum yazılımının güvenilir mühendisliği için yöntemler ve yaklaşımlar geliştirmeye, programlardaki kuantum hatalarını düşük maliyetlerle keşfetmeye odaklanır. Klasik test yaklaşımları kuantum bilgisayarlarında çalışan karmaşık kuantum programları için ölçeklenebilen sistematik ve otomatik test yaklaşımları sunmamaktadır [4]. QC'nin uzmanlaşmış özellikleri (üst üste binme ve dolanıklık) nedeniyle test aşamasındaki zorlukları elimine edebilmek için önemli çalışmalar yapılmış ve çeşitli test araçları geliştirilmiştir. [6]. Bunların arasında bulanıklık test, mutasyon analizi, arama tabanlı test, kombinasyonel test ile birlikte kuantum giriş çıkış testi (Quantum Input Output testing – Quito)[8] aracı yer almaktadır. Bir kuantum programını bir kara kutu olarak ele almak ve bir kuantum programı yürütmesinin sonunda durumları okumak bile olasılıksal çıktılarla sonuçlanır. Çıktıları değerlendirmek için, bir kuantum programını birden çok kez yürütmeye ve ilgili istatistiksel testleri uygulamaya başvurmak gerekir. Bu amaçla kuantum programlarının istatistiksel yaklaşımlarla yapılmış çalışmalar mevcuttur [7]. Ayrıca, birçok kuantum programının önceden tanımlanmış test araçları yoktur, bu da kuantum programlarının doğruluğunu belirlemede başka bir zorluktur. Bu amaçla, kuantum programlarını test etmek için metamorfik test [8] yaklaşımları, bu yönde değerli çalışmalardır.



Şekil 1. Kuantum Test Araçlarının Tipik İşleyişi *(Typical Operation of Quantum Test Tools)*

YÖNTEMLER VE KAPSAMLAR (METHODS AND SCOPE)

1. Kapsayıcı Kriterler *(Inclusive Criteria)*

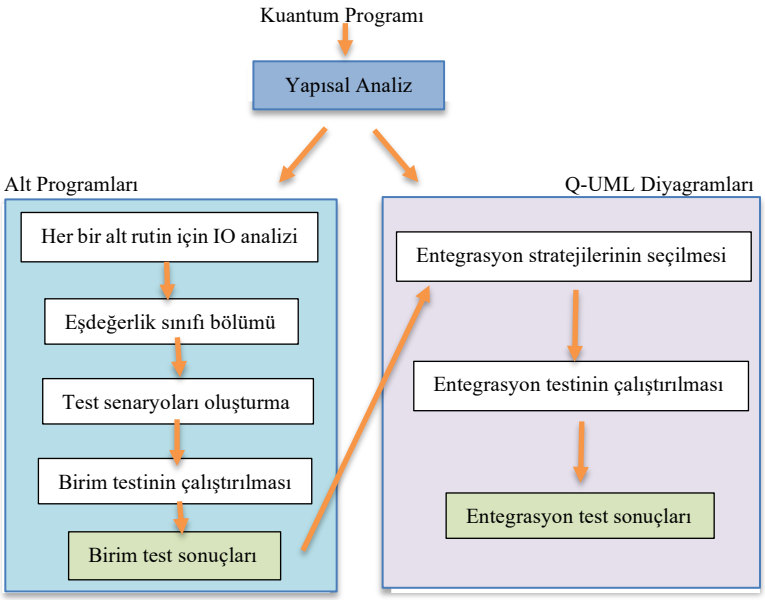
Test edilecek programın, test araçları veya yöntemleri ile tam kapsamlı ve uyumlu bir şekilde değerlendirilebilmesi için kriterler belirlenir. Kuantum programlarında fonksiyon ve doğruluk analizleri yapılırken, farklı hataları veya potansiyel problemleri tespit etmek için çeşitli yöntemler bir araya getirilir. Kapsayıcı kriterler, genellikle bir kuantum yazılımının farklı durumlarını (örneğin, süperpozisyon, dolanıklık, ölçüm süreçleri) kapsayacak şekilde bir test senaryosu sunmayı amaçlar.

Şekil 2[9]'de kuantum programlarının test edilmesine ilişkin akış diyagramı yer almaktadır. Bu diyagrama göre kuantum programı girdi olarak verildiğinde öncellikle yapısal analizi gerçekleştirilir. Yapısal analiz, kuantum programlarının alt rutin ve organizasyonlarını tespit etmek için gereklidir. Bu aşamada programın kuantum özelliklerini modellemek için Kuantum Yazılım Modelleme Dili (Quantum Software Modeling Language – Q-UML)[10] kullanılır. Q-UML kuantum yazılımını düzgün bir şekilde modellemesine olanak tanıyan Birleşik Modelleme Dili'nin (Unified Modeling Language - UML)[10] bir uzantısıdır. Her bir alt rutin üzerinde birim testi ve ardından tüm program üzerinde entegrasyon testi yapılır.

Alt rutin testleri için öncelikle girdi ve çıktı analizi yapılır. Kullanıcının atadığı değişkenler girdi olarak kabul edilirken program çalıştırıldıktan sonra elde edilen kullanıcının ilgilendiği değişkenler çıktı olarak kabul edilir. Bu aşamanın ardından kuantum girdilerini bölümlendirmek için eşdeğerlilik sınıfı (Equivalence Classes) bölümleme testi uygulanır. Kübitlerin alt kümesi olarak tanımlanan değişkenlerinin taşıdığı değerler kuantum durumları olarak tanımlanır. Temel de üç kuantum durumu vardır bunlar süperpozisyon, karma ve klasik durumdur. Bölümleme kriteri kuantum değişkeninin durum türlerine bağlıdır. Bu bölümlemeye Klasik-Üst üste binme-Karma Bölme(Classical-Superposition-Mixed Partition- CSMP)[9] denir. Bununla birlikte karmaşık durumda çalışmayan kuantum durumları için klasik-süperpozisyon bölümü (Classical-Superposition Partition - CSP)[9] kriteri uygulanır. Süperpozisyon durumundan doğan dolanıklığında kapsanması gerekmektedir. Bunun için dolanıklık kapsamı (Entanglement Coverage - EntC)[9] ve süperpozisyon her bir kübit için üretilir. Öte yandan birden çok girdi değişkenine sahip olan kuantum programları için klasik kapsam kritelerinden, Tüm Kombinasyon Kapsamı (All

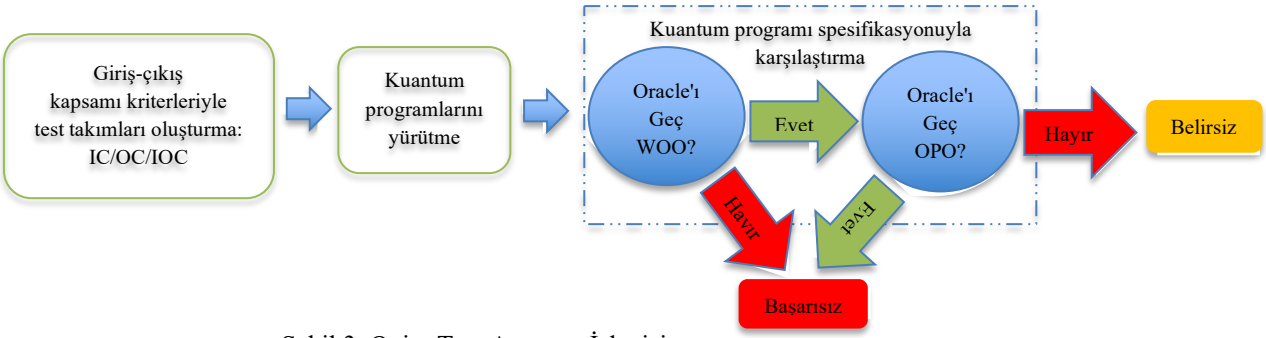
Combination Coverage - ACoC)[9], Her Seçim Kapsamı (Each Choice Coverage - ECC)[9], Çiftler Halinde Kapsam (Pair-Wise Coverage - PWC)[9] ve Temel Seçim Kapsamı (Base Choice Coverage-BCC)[9] kuantum programlarına uyarlanabilir. Kapsam kriterleri uygulandıktan sonra test senaryoları oluşturulur. Test senaryosu temelde kuantum girdilerinin oluşturulması ve sonuçların tespit edilmesi olarak iki aşamadan oluşur. Klasik testlerden farkı beklenen çıktı kombinasyonları yerine İstatistik Tabanlı Algılama (Statistics Based Detection - SBD)[9], Kuantum Çalışma Zamanı İddiaları (Quantum Runtime Assertions - QRA)[9] gibi teknikler kullanılır. Klasik testlerdeki gibi girdi durumları altında çıktıların elde edilmesi kuantum programları için yeterli doğruluk sunmaz. Hedef program üniter bir yapıdaysa ek birimsellik (Unitarity) kontrolü yapılır.

Alt programlar arasında oluşabilecek hataları elimine etmek için son olarak entegrasyon testine tabi tutulur. Bu aşamada Q-UML rehber olarak kullanılır. Klasik entegrasyon testlerinde olduğu gibi entegrasyon sırasını belirlemek için bağımlılık grafiği üzerinden topolojik olarak sıralama yapılır. Entegrasyon esnasında tanımlanmamış rutinler kullanılması gerekiyorsa test ikilisi kullanılır. Bu teknik kuantum programlarını Kahinlerle (Oracle) test etmek için uygundur [9].



Şekil 2. Çok Alt Rutinli Kuantum Programının Test Edilmesine İlişkin Genel Süreç Diyagramı *(General Process Diagram for Testing a Quantum Program with Multiple Subroutines)*

Programın test kalitesinin objektif bir ölçüsünü sağlamak için Quito tercih edilebilir. Bu araç özelleştirilebilir test senaryoları sunarak, programın hem kuantum (kapı yapılandırmaları) hem de klasik bileşenleri (algoritmalar) arasında tutarlılığı kontrol eder. Geniş bir test kapsama alanı sağlamak için metamorfik test stratejileriyle de uyumludur. Quito, genellikle kuantum devrelerinin beklenen özellikleri karşılayıp karşılamadığını doğrulamak için kullanılır. Dekoherans (Bağlı durumun kopması) etkiler, devre derinliği kısıtlamaları ve kuantum kapısı hataları gibi kuantum bilişimine özgü sorunları tespitinde de faydalanılmaktadır [2]. Şekil 3[2,11,12]'de Quito Test aracının işleyiş şeması verilmiştir. İlk olarak kapsayıcı kriteler ile test takımları oluşturulur. Bir test takımında her geçerli girdi ve çıktı için bir test vardır. Statik olarak oluşturulan test takımı girdi kapsamını (Input Coverage – IC) oluştururken çıktı kapsamı (Output Coverage – OC) ve girdi-çıktı kapsamı (Input - Output Coverage – IOC) statik olarak elde edilemez[11]. Kapsayıcı kritelerin kullanılması çok alt rutinli programlarda üstel karmaşıklığa sebep olması nedeniyle tercih edilmez. Girdi olarak verilen programın spesifikasyonları mevcutsa oluşturulan test paketleri iki test kahini kullanılarak değerlendirilir. Yanlış Çıktı Kahini (Wrong Output Oracle - WOO)[11] ve Çıktı Olasılık Kahini (Output Probability Oracle - OPO)[11]'dir. WOO, Test girdisi için döndürülen her çıktı değerinin geçerli olup olmadığı kontrol ederken, OPO ise her IO çifti için oluşturulan test takımları (Test Set - TS) arasında tek örnek Wilcoxon işaretli sıralama testi gerçekleştirir. Hipotez reddedilirse belirsiz, kabul edilirse başarısızlık bildirir.



Şekil 3. Quito Test Aracının İşleyişi *(How the Quito Test Tool Works)*

Çok alt rutinli kuantum programları için kuantum alt programlarının birleşimini destekleyerek alt rutinlerinin test gereksinimlerini belirlemeye ve yönetmeye yardımcı olması amacıyla çoklu alt rutin Q# programları için bir test aracı (QsharpTester)[9] geliştirilmiştir. Birincil programlama dili olarak Q# dili ile yazılan kuantum programlarının test edilmesini kolaylaştırır. Başlıca odak noktası, hibrit algoritmaların hem kuantum hem de klasik bileşenlerinin düzgün bir şekilde test edilmesini sağlamaktır. [9] yapılan çalışmada yedi orjinal doğru program ve bunların dört mutasyon türüne sahip 244 adet hatalı mutasyon içeren bir dizi Q# karşılaştırma programı kullanılmış ve deney sonucunda neredeyse tüm hatalı sonuçlar tespit edilmiştir. QSharpTester test çerçevesinin scaffold, silQ ve Qiskit gibi kuantum programlama dillerinde de uygulanabileceği önerilmiştir [9].

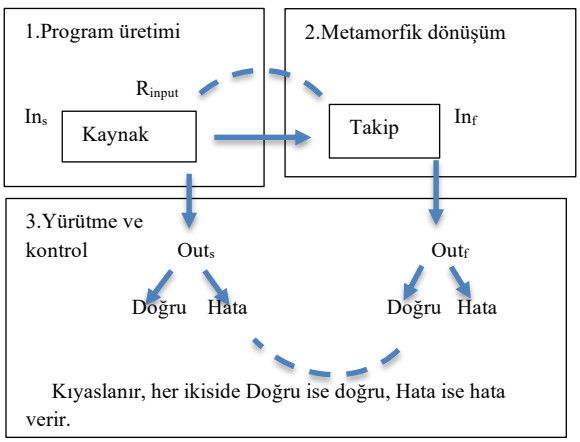
2. Test Teknikleri *(Testing Techniques)*

Kuantum yazılım test teknikleri yapı ve işleyiş bakımından çeşitlilik gösterir. Karmaşık kuantum davranışlarının doğruluğunu analiz etmek için metamorfik test tekniklerinden MorphQ kullanımı tercih edilirken, kuantum devrelerinin doğru çalışıp çalışmadığını denetlemek için mutasyon analizinden QMutPy ve Muskit test araçları veya kombinasyonel testlerden QuCAT test aracının kullanımı önerilir. QDiff gibi araçlarsa, farklı kuantum platformları arasında uyum kontrolü sağlayarak platformdan bağımsız testlerin gerçekleştirilmesine olanak sağlar. Gerçekleştirilmek istenen test çalışmasının ihtiyaçları doğrultusunda birden fazla araç tercih edilebilir. Makalemizin bu bölümünde bahsi geçen teknik ve araçlara yer verilmiştir.

2.1. Metamorfik Testler *(Metamorphic Tests)*

Metamorfik testler, test vakaları arasındaki beklenen dönüşümlerin tutarlı ve mantıksal sonuçlar üretmesini sağlayarak kuantum algoritmalarının doğruluğunu test eder. Her bir çıktının doğruluğunu kontrol etmek yerine, girdilere karşılık gelen çıktılar arasındaki ilişkilerin beklendiği gibi olup olmadığına odaklanır. Bu ilişkiler "metamorfik ilişkiler" olarak bilinir. Metamorfik test kavramını, kuantum programlarına uygulamak için MorphQ aracı kullanılmaktadır. On metamorfik ilişkiden oluşan MorphQ; devreyi değiştiren devre dönüşümleri, temsil dönüşümleri ve yürütme dönüşümlerinden oluşur[13].

Şekil 4[13]'de MorphQ ve üç ana adımına ilişkin bir genel bakış sunulmaktadır. İlk olarak, bir program üreticisi kaynak program olarak adlandırılan bir başlangıç kuantum programı (In_s) oluşturur. Ardından, bir dizi metamorfik program dönüşümü uygulayarak, yaklaşım kaynak programla belirli bir ilişki içinde olan bir takip programı (In_f) üretir. Son olarak, yaklaşım iki programı çalıştırır ve davranışlarının beklenen çıktı (Out_s ve Out_f) ilişkisine uyup uymadığını kontrol eder. Ana döngü, sürekli olarak yeni kaynak (R_{input}) çiftleri oluşturur ve kontrolünü sağlar [13] .

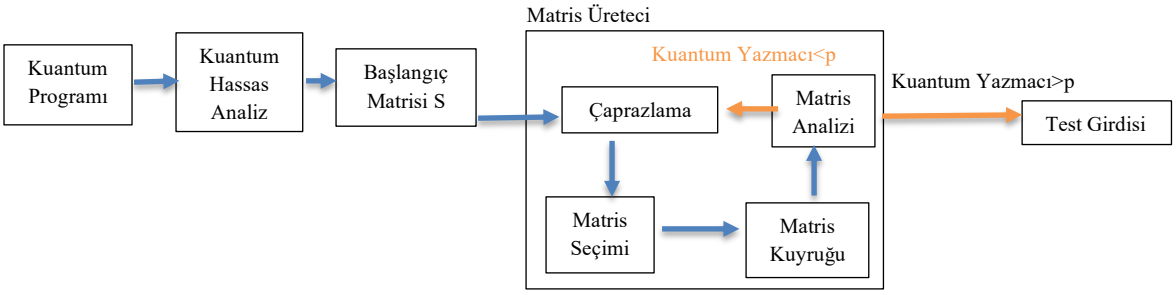


Şekil 4. MorphQ yaklaşımına genel bakış *(Overview of the MorphQ approach)*

2.2. Fuzz Testi *(Fuzz Test)*

Bulanık test (fuzz testing), klasik sistemlerde bir programa rastgele, beklenmeyen veya geçersiz veriler girerek güvenlik açıklarını ve hataları belirlemek için kullanılan bir yazılım test tekniğidir. Bu yöntem, yazılımın beklenmedik girdilere dayanabilmesini sağlamaya yardımcı olduğu için karmaşık girdi işleme özelliğine sahip programlar için kullanışlıdır. Kuantum sistemlerindeki belirsizlik, Kuantum yazılım testlerinde de olasılıksal sonuçların alınması gibi kısıtlamalara yol açar. Bu durum, fuzz testlerinde sağlıklı sonuçların üretilmesi için geleneksel 'beklenen çıktı' yöntemi yerine olasılıksal hata tespit algoritmalarının tercih edilmesini sağlar. QuanFuzz, kuantuma duyarlı bilgilerin tanımlanıp kuantum kayıtlarının durumunu değiştirerek kapsamı en üst düzeye çıkarmayı amaçlayan GreyBox bulanık test aracını kullanır[14]. Başlangıç matrislerini mutasyona uğratarak kuantum duyarlı dalları seçmek için yüksek ağırlık değerli matrisleri üretir. Geleneksel test yöntemleriyle karşılaştırıldığında QuanFuzz, özellikle kuantum duyarlı dallarda kapsam %20 – 60 oranında artırmaktadır[14].

Şekil 5[14]'de QuanFuzz'un iş akışı gösterilmiştir. Başlangıçta girdi olarak verilen kuantum programının hassas analizi yapılır. Analiz sonucuna göre başlangıç matrisi oluşturulur. Matris sayısını artırmak için kuantum kapıları kullanılarak matrisler üretilir. Üretilen matrisler olasılık ağırlıklarına göre değerlendirilir. Seçilen matrisler, matris analizlerinin yapılması için sıraya alınır. Kuantum yazmacı olasılık ağırlığı eşik değeri p'den büyük olanlar test girdisi olarak seçilir değilse çaprazlama aşamasına geri döndürülür [14].



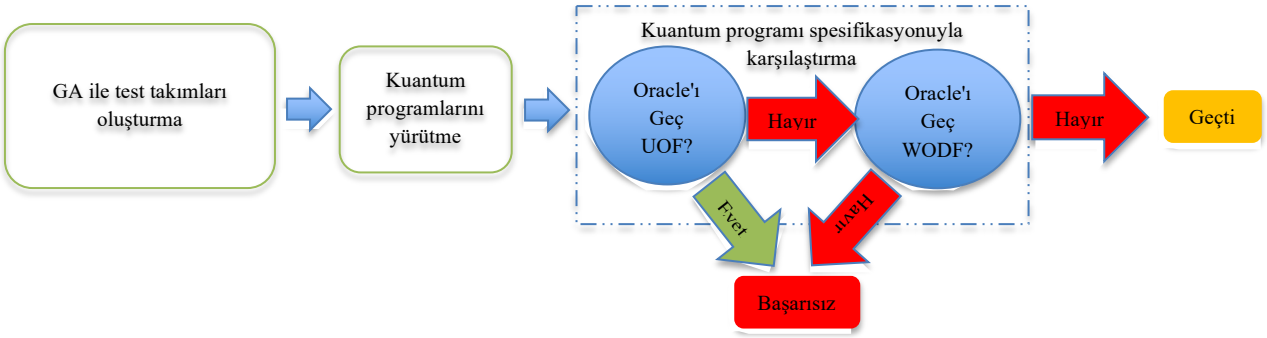
Şekil 5. QuanFuzz genel iş akışı *(QuanFuzz general workflow)*

2.3. Arama Tabanlı Test *(Search Based Testing)*

Yazılım test serilerinde genellikle, genetik uygulamalar ve arama tekniklerinden oluşan çeşitli senaryolar kullanılır. Arama tabanlı testler, yazılımın performans göstergeleri ve hataların bulunması için en iyi test durumlarını bulma işlemlerini içerir. Arama tabanlı test, klasik yazılımlarda olduğu gibi kuantum programlarında da mevcuttur. Testler, genellikle kuantum simülatörleri veya kuantum bilgisayarlarında çalıştırılır. Bu, test sürecini daha karmaşık ve maliyetli hale getirmesiyle beraber kuantum yazılımının doğası gereği olasılıksal çıktılarla sonuçlanmasına neden olur. Bu alanda Kuantum Arama Tabanlı Test (Quantum Search-Based Testing - QuSBT)[11] ve Çok Amaçlı Arama Tabanlı Yaklaşım (Multi-Objective Search-Based Approach - MutTG)[11] gibi araçlar kullanılmaktadır.

QuSBT bir genetik algoritma (Genetic Algorithm - GA)[11] kullanarak, belirli kuantum programları için olası hata durumlarını belirlemeye çalışır. Bu araç, programın öngörülen çalışmasına göre hata senaryoları oluşturup başarısız test durumlarının bulunması için kullanılır. QuSBT'nin temel amacı, kuantum programlarının hata tespitini optimize etmektir. [15] yapılan çalışmada, QuSBT'yi beş kuantum programının on hatalı sürümü ile test edip, ortalama olarak test vakalarının en az %50'sinin başarısız olduğu test takımları üretmeyi başarmıştır. [11] yapılan çalışmada ise QuSBT 30 hatalı kuantum programı ile değerlendirilmiş, programların %87'sinde Rastgele Arama (Random Search - RS)'dan daha iyi performans göstermiştir. Şekil 6[11,15]'da rastgele aramanın baz alındığı QuSBT aracının işleyiş şeması sunulmuştur. İlk olarak, test edilen kuantum programının giriş bilgileri (giriş ve çıkış kübitlerinin listesi, toplam kübit sayısı ve Program Spesifikasyonu (Program Specification - PS)[11] sağlanır. Ardından sağlanan giriş bilgileri doğrultusunda GA yapılandırmaları gerçekleştirilerek test takımları oluşturulur. Programın yürütme aşamasında ise programın giriş alanı üzerinde yapılan arama, tam sayı değişkenleri ile temsil edilir. Aramanın amacı, başarısız testlerin sayısını en üst düzeye çıkaran bir atama bulmaktır [15].

Test yürütme sonucunun doğruluğu PS karşılaştırmaları ile yapılır. İki hata türü açısından Beklenmeyen Çıkış Hatası (Unexpected Output Failure - UOF)[15] türünde bir hatanın meydana gelip gelmediğini, yani PS'ye göre meydana gelmemesi gereken bir çıktının üretilip üretilmediğini kontrol eder; durum buysa, sonuç başarısız olur ve Yanlış Çıkış Dağıtım Hatası (Wrong Output Distribution Failure - WODF)[15] için değerlendirme yapılmaz. Aksi durumda ise PS durumunda beklenen dağılımın izlenip izlenilmediğini WODF'ye göre değerlendirir. Bu işlem verilen veya varsayılan önem düzeyini kullanarak Pearson'ın ki-kare testi ile bir uyum iyiliği testi gerçekleştirerek yapılır. İstatistiksel test önemli bir fark gösteriyorsa başarısız, aksi takdirde geçti olarak sonuç üretir [15].



Şekil 6. QuSBT Test Aracının İşleyişi *(How the QuSBT Test Tool Works)*

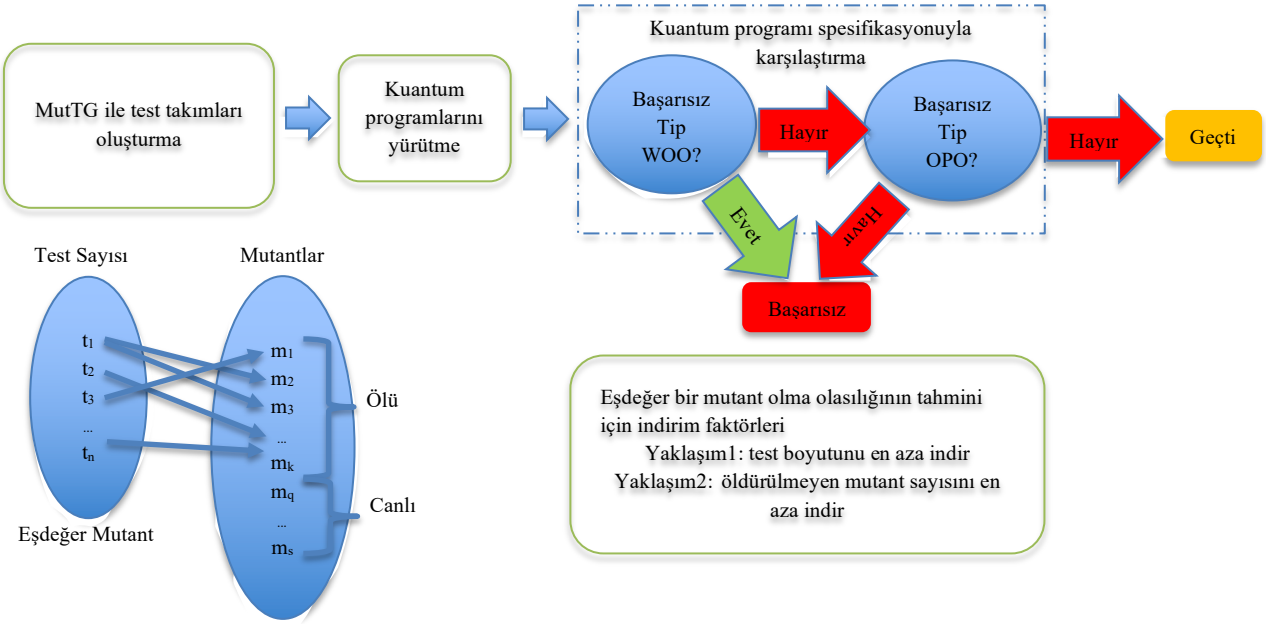
Bir diğeri ise çok amaçlı arama tabanlı yaklaşım (Multi-Objective Search-Based Approach – MutTG)[11]'dır. Kuantum programları için standart tabanlı bir test oluşturur. Şekil 7[11,16]'de MutTG aracının işleyiş şeması sunulmuştur. Kuantum programları için mutasyon analizi, bir kuantum programı (Quantum Program - QP) verildiğinde mutant (M), devreye örneğin, yanlış bir kapı kullanımı gibi söz dizimsel bir hata sokularak elde edilen QP'nin biraz farklı bir versiyonu olarak üretilir. Bir testte $(t)M$'nin yürütülmesi, orijinal program QP üzerinde t'nin yürütülmesinden önemli ölçüde farklı sonuçlar üretirse, bir mutantın öldürüldüğü söylenir. Dolayısıyla, bir mutantın öldürülüp öldürülmediğini kontrol etmek için onu PS ile karşılaştırabiliriz. Özetle mutant yürütülmesi iki olası başarısızlıktan birine yol açarsa öldürülür.

Programın yürütülmesi aşamasında değişkenler tanımlanır. Arama sırasında, Muts'taki hangi mutantların öldürüldüğünü kontrol etmek için karşılık gelen testler yürütülür. Arama testlerinde öldürülen mutantlar (Killed Mutant – KM)[16] sözlüğüne, bunların dışındaki mutantlar (Not Killed Mutant – NKM)[16] sözlüğüne girdi olarak alınır. Bu işlemden sonra hedef fonksiyonlar değerlendirilir. İlk hedef, test takımı boyutunu en aza indirmektir. Bunun için öldürülmedi puanı (Not Killed Score - NKS)[16] isimli uygunluk fonksiyonu Denklem. [1,15] kullanılır.

$$= \sum_{M \in Muts} \frac{f_{notkilled}(v)}{notKilledScore(M, v, KM, NKM)} \quad [1]$$

Bu fonksiyon mutant M'nin öldürülemez olma olasılığını belirten [0, 1]'de tanımlanan bir endekstir. En iyi durum "0" en kötü durum "1" olarak derecelendirilir. Mutant M, mevcut bireyin bir testi (ilk durum) tarafından öldürülürse, "öldürülmemiş puanı" 0'dır. Mutant M mevcut bireyin bir testi tarafından öldürülmezse, ancak daha önce oluşturulan testlerden biri tarafından öldürülmüşse (ikinci durum), "öldürülmemiş puanı" 1'dir. Mutant M mevcut birey tarafından

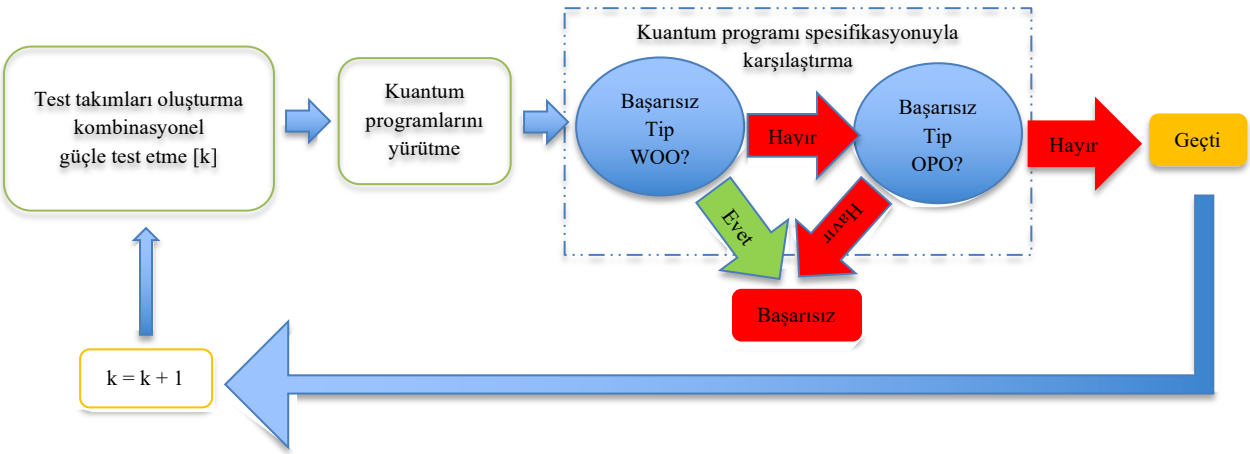
öldürülmemiş, ancak daha önce oluşturulan testler tarafından öldürülebilir olduğu da bilinmiyorsa (üçüncü durum), "öldürülmemiş puanı" belirli bir indirim faktörünün en kötü değeri 1'den düşürülür. Bu faktör, M'yi öldürmeyen testlerin olası girdilerin toplam sayısına oranıdır. Fonksiyon, M mutantını öldürmeyen ne kadar çok test denersek, M'nin öldürülebilir olmayan eşdeğer bir mutant olma olasılığı o kadar artar. Bu indirim faktörünü kullanarak, aslında eşdeğer olan mutantları öldürmeye devam etmekten kaçınma amaçlanmıştır [16]. Test yürütme sonucunun doğruluğu PS karşılaştırmaları ile yapılır.



Şekil 7. MutTG Test Aracının İşleyişi *(How the MutTG Test Tool Works)*

2.4. Kombinasyonel Test *(Combinational Testing)*

Kombinasyonel test, girdi kombinasyonlarını sistematik bir şekilde kullanarak kuantum programlarındaki hataları tespit eder. Kuantum programlarının doğruluğunu artırmayı hedefler[12]. Bu amaçla geliştirilmiş olan Quantum Kombinasyonel Test(Quantum Combinatorial Testing- QuCAT)[11], kuantum programlarının giriş ve çıkış yöntemlerini analiz eder. İki şekilde kullanılır; bunlardan ilki, kullanıcı sabit bir k değeri belirler ve kombinasyonel test takımı oluşturulur. Bir diğeri ise bir hata bulunana veya maksimum k değerine ulaşılan kadar k'nın ilk değeri 2 den başlatılarak artırımlı olarak test takımları oluşturulmasıdır. Maksimum k değerine ulaşana kadar PS geçti olarak sonuçlanır. Şekil 8 [11,17]'de rastgele aramanın baz alındığı QuCAT test aracının işleyiş şeması gösterilmiştir. Testlerin başarı durumu, diğer test araçlarında olduğu gibi PS karşılaştırmaları ile yapılır [11].

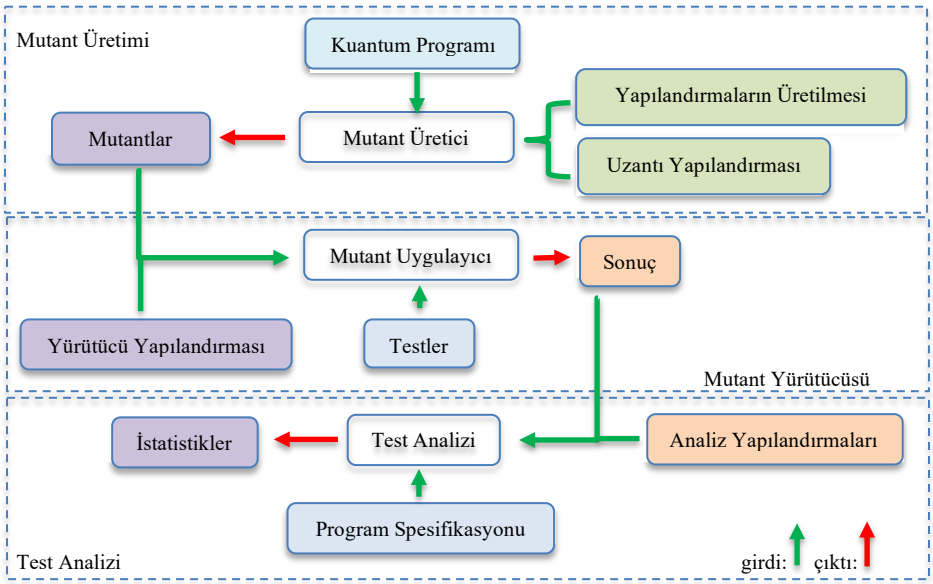


Şekil 8. QuCAT Test Aracının İşleyişi *(How QuCAT Test Tool Works)*

2.5. Kuantum Mutasyon Analizi *(Quantum Mutation Analysis)*

Kuantum programlama bağlamında, test teknikleriyle üretilen test vakalarının kalitesini değerlendirmek için hata depoları ve kıyaslama programları yeterli değildir. Bu duruma alternatif bir çözüm sunmak için mutasyon analizi kullanılmaktadır. Yazılım test süreçlerinde, özellikle de kuantum programları için test kalitesini değerlendirmek amacıyla kullanılan bir yöntemdir. Bir mutantı test vakasında yürüttüğümüzde, QP'nin program spesifikasyonuna (beklenen çıktı dahil) bağlı olarak önceden tanımlanmış kriterlere göre (örneğin, belirli bir girdi için yanlış bir çıktı gözlemlenmesi) başarısız olursa, bu mutantın öldürüldüğü anlamına gelir. Mutasyon skoru genellikle bir test takımının kalitesini değerlendirmek için kullanılır. Burada, test takımı tarafından öldürülen mutantların sayısını toplam mutant sayısından hesaplanır; eşdeğer mutantların sayısı biliniyorsa, hesaplama sırasında toplam mutant sayısından çıkarılır. Bu bağlamda iki önemli araç bulunmaktadır. İlki kuantum yazılım testi için bir mutasyon analizi aracı Muskit'tir[11]. Kuantum devrelerinde bulunan kapıları (Gates) hedef alarak çeşitli mutasyon operatörleri tanımlar. Muskit, iki ana bileşene sahiptir: Mutant Üretici(Mutant Producer) ve Mutant Yürütücü(Mutant Executor). Mutant üretici, belirli bir kuantum programı için mutantlar üretirken, mutant yürütücü, bu mutantlar üzerinde testleri çalıştırarak sonuçları değerlendirir.

Şekil 9[11,18]'da işleyiş diyagramı sunulmuştur. Muskit, kuantum devrelerinin mutasyona uğramış özelliklerine odaklanır. Bu amaçla, iki kavram tanımlar: kapı numarası ve konum. Kapı numarası, silme veya değiştirme operatörleri aracılığıyla mutasyona uğratmak istediğimiz bir kuantum devresindeki belirli bir kapıyı (örneğin, h, G1'dir) ifade eder. Konum ise kuantum devresinde mutasyona uğratmak istediğimiz yere, yeni bir kapı ekleyerek belirlenen yeri ifade eder. Üç operatör türüne göre mutasyon operatörü tanımlanır. Bunlar; Kapı Ekleme(Add Gate - AG)[18], Kapı Kaldırma(Remove Gate - RemG)[18] ve Kapı Değiştirmedir(Replace Gate - RepG)[18]. Giriş kübit sayısına göre kullanıcı bu operatörlerden bir veya daha fazlasını seçerek mutant üretimini kontrol eder. Mutant yöneticisi girdi olarak mutantlar ve test vakalarını alır, mutantlar üzerinde test vakalarını yürütür ve sonuç üretir. QP'lerin olasılıksal doğası göz önüne alındığında, her mutant belirtilen sayıda test vakasıyla yürütülür. Test Analizcisi(Test Analyzer), kullanıcının test değerlendirme kriterlerini uygular. Burada kullanıcı üç şeyi belirtir: ilki seçilen p değeri düzeyi (örneğin 0,05), diğeri girdi olarak kullanılan kübit kimlikleri ve son olarak ölçülecek kübit kimlikleridir. Bir program spesifikasyonu (örneğin, beklenen çıktılar ve her girdiye karşılık gelen olasılıklar) ve mutant yürütücü'den test sonuçları verildiğinde, test analizcisi bir mutantın bir test vakası tarafından öldürülüp öldürülmediğini söyleyebilir [18].



Şekil 9. Muskit Test Aracının İşleyişi *(How Muskit Test Vehicle Works)*

Bir diğer aracı ise QMutPy'dır. MutPy adlı açık kaynaklı mutasyon aracının kuantum programları için genişletilmiş bir versiyonudur[19]. QMutPy, kuantum ölçümleri ve kapıları temel alarak yeni mutasyon operatörleri oluşturur. IBM'in Kuantum Bilgi Yazılım Kit (Quantum Information Software Kit –Qiskit)[19] kütüphanesi kullanılarak yazılmış gerçek kuantum programları üzerinde testler gerçekleştirilmiştir. QMutPy'nin iş akışı MutPy benzer şekilde dört ana adımdan oluşur. Bir Python programı (P), onun test takımı (T) ve bir mutasyon operatörü kümesi (M) verildiğinde, QMutPy öncelikle P'nin kaynak kodunu ve test takımını yükler ardından T'yi orijinal (değiştirilmemiş) kaynak kodunda yürütür ve M'yi uygular. Son olarakta P'nin tüm mutant sürümlerini üreterek T'yi her mutant sürümünde yürütür [19].

[20]'de, IBM'in Qiskit kütüphanesinde yazılmış, kuantum mutasyon operatörleri setinin 11 QP için toplam 696 mutant üretilmiş ve bunların 325'i (%46,7) programların test paketleri tarafından öldürülmüştür. Öldürülmeyen mutantlar ya test paketlerine kadar hayatta kalmış (307, %44,1) ya da test paketleri tarafından kullanılmamış (%0,3) veya zaman aşımına uğramıştır (62, %8,9). Elde edilen sonuçlar bu alanda QMutPy gibi araçların önemli bir rol oynadığını göstermektedir. Bunun yanı sıra Muskit test aracının şu anda eşdeğer mutantları tespit edemediği. Bunun başlıca nedeninin, mevcut bir gövdenin olmadığı belirtilmiştir[18].

2.6. Kuantum Platform Testi *(Quantum Platform Test)*

Kuantum mekaniğinin fiziksel özellikleri nedeniyle, kübitler ve kuantum kapıları klasik bitlerden ve kapı mantığından temelde farklıdır. Birçok kuantum yazılım yığını, altta yatan fiziksel ve matematiksel karmaşıklıkları soyutlayarak kuantum programlama için kullanıcı dostu üst düzey diller geliştirmiştir. Bunlar arasında CirQ, PyQuil ve Qiskit dilleri sayılabilir. Kuantum yazılım yığını (Quantum Software Stack – QSS)[18]; API'ler aracılığıyla verilen kuantum algoritmasını devre düzeyinde dönüştüren ve optimize eden bir derleyici ile ortaya çıkan kapıları klasik aygıtlarda simüle eden veya doğrudan kuantum donanımında yürüten bir arka uç yürütücü içerir. Programları derleyen ve optimize eden Qiskit Terra, yüksek performanslı gürültülü simülasyonları destekleyen Qiskit Aer, hata düzeltme, gürültü karakterizasyonu ve donanım doğrulaması için Qiskit Ignis, bir geliştiricinin bir kuantum algoritmasını veya uygulamayı ifade etmesine yardımcı olan Qiskit Aqua olmak üzere dört bileşenden oluşan Qiskit örnek verilebilir.

QS'i test etmeyi zorlaştıran başlıca etkenler mevcuttur. İlki çok platformlu program çevirmenidir(Multi-Platform Program Translator). Kuantum programları genellikle yeni programlama dillerinde yazılır veya mevcut dillerin üstündeki API'ler kullanılarak ifade edilir. Standartlaştırılmış ara gösterimlerin yokluğu, platformlar arası test için bir zorluk oluşturur. Bir diğer zorluk, kuantum program üretimidir. Platform testi çok sayıda program gerektirir. Ancak günümüzde yalnızca birkaç gerçek kuantum program örneği mevcuttur. Son olarak Çok Değişkenli İkili Dağılım Karşılaştırması(Multivariate Binary Distribution Comparison), kuantum ölçümlerinin olasılıksal doğası, çıktıların dağılımlarla temsil edilmesiyle karmaşıklık oluşturur. Bu zorlukla ilgili olarak kuantum hata ayıklamada Kolmogorov-Smirnov (KS) veya Çapraz Entropi (Cross Entropy) testleri kullanılarak yapılan ön çalışmalarla birlikte araştırmalar yürütülmüştür [21]. QSS testlerinde meydana gelen zorlukları aşmak adına çeşitli platform testleri geliştirilmiştir. Quantum Yazılım Yığınlarının Farklı Testleri (Differential Testing of Quantum Software Stacks - QDiff)[21] bunlardan biridir. QDiff, girdi olarak kuantum programı alır ve bir çift tanık programla (yani, aynı davranışı üretmesi beklenen mantıksal olarak eşdeğer program) olası hataları bildirir.

Bir başka zorluk, kuantum simülatörlerinden veya donanımdan gelen ölçümlerin yorumlanmasıdır. Kuantum ölçüm sonuçları olasılıksal değerler üretir. Bunun için içsel gürültüyü (örneğin, donanım kapısı hataları, okuma hataları ve dekoherans hataları) hesaba katmalı ve gözlemlenen sapmanın anlamlı bir kararsızlık olarak kabul edilebilecek kadar önemli olup olmadığı belirlenmelidir. Bu, her kuantum kapısının matematiksel olarak bir üniter matrislerle temsil edilebileceği anlayışına dayanır; bir kapı dizisi esasen üniter matrislerinin çarpımına karşılık gelir ve bu da bir üniter matris üretir. Bu nedenle, iki dizi aynı üniter matrisi üretirse, bir kapı dizisi diğerine anlamsal olarak eşdeğerdir. Aynı üniter matrisi üretmesi garantili farklı kapı dizileri üretmek için yedi kapı dönüşüm kuralından yararlanılır. Bu diziler esasen aynı davranışı üretmesi beklenen mantıksal olarak eşdeğer program varyantlarını tanımlar. Üretilen birden fazla mantıksal eşdeğer varyantlardan çalıştırılmaya değer olan devrelerin bir alt kümesi seçilir. Son olarak mantıksal eşdeğer devrelerin yürütmelerini karşılaştırmak için QDiff, güvenilir karşılaştırma için ne kadar ölçümün gerektiğini belirler. Ölçüm sayısını tahmin etmek için yakınlık testini uygular. Ardından gereken sayıda ölçümü gerçekleştirir. İki ölçüm kümesini karşılaştırmak için QDiff, dağıtım karşılaştırma

yöntemlerini kullanır. KS testine ve çapraz entropiye dayalı iki yöntem desteklenmektedir [21].

3. Kuantum Programları İçin Hata Tespiti *(Error Detection for Quantum Programs)*

Kuantum programlamadaki güncel araştırmalar esas olarak sorun analizi, dil tasarımı ve uygulamaya odaklanmaktadır. Kuantum yazılım geliştirme süreçlerini iyileştirmek, araştırmacılara gerçek hataları inceleme imkanı sağlar. Fakat, kuantum programlarındaki hataların ayıklanması konusu kuantum programlama paradigmasında çok az ilgi görmüştür. Kuantum programlamada tanıtılan üst üste binme, dolanıklık gibi belirli özellikler, kuantum programlarındaki hataları bulmayı zorlaştırmaktadır. Kuantum yazılımlarını hata ayıklamak ve test etmek için çeşitli yaklaşımlar önerilmiştir [22].

İlki Qbugs'tır. Qbugs, kuantum yazılım testinde ve hata ayıklamada kullanılmak üzere tasarlanmış bir altyapı sağlar. Farklı kuantum programlama dilleri (Q#, OpenQASM, CirQ, Quipper ve Scaffold) için desteklenir. QBugs'ın bir prototipini oluşturmak için kuantum çerçeve depolarında bulunan kuantum algoritmalarının açık kaynaklı uygulamaları kullanılmıştır. Bunlar; ProjectQ'nun çerçeve deposu, QiskitAqua'nın deposu ve O'Reilly'nin Kuantum Bilgisayarları Programlama" kitabının deposudur. Hangi hataların bildirildiğini otomatik olarak tespit edip gerekli düzeltmenin yapılması için Defects4J veritabanı ve BugsJS veritabanı kullanılmıştır [23].

Bir diğeri ise Bugs4Q'dur. Bu araç Qiskit programlarındaki yeniden üretilebilir hataları toplar ve kuantum yazılım testleri için test durumlarını indirmeyi ve çalıştırmayı destekler. Her gerçek hata ve karşılık gelen düzeltmeler erişime açıktır. Qiskit'in GitHub'daki mevcut hatalarının neredeyse tamamını toplar ve dört popüler Qiskit öğesinin (Terra, Aer, Ignis ve Aqua) gerçek zamanlı olarak günceller. Ayrıca, bu programlar orijinal olarak mevcut test durumları ve yeniden üretim desteği olan hatalar haricinde ayrı ayrı sıralanır ve filtrelenir. Bugs4Q, izole edilmiş hataların deneysel değerlendirmesi için mevcut hataları sınıflandırmak üzere hata türlerinin analizini içeren bir veritabanı sağlaması yönüyle kuantum programlarındaki gerçek hataların bir kataloğudur [22].

Tablo 1.'de Test araç ve tekniklerinin avantaj, dezavantaj ve performans değerlendirmeleri özetlenmiştir. *(Advantages, disadvantages and performance evaluations of testing tools and techniques are summarized.)*

Kategori	Araç	Avantajlar	Dezavantajlar	Performans Değerlendirmesi	Referanslar
Kapsayıcı Kriter	Quito	Kuantum algoritmaları için özelleştirilmiş test çerçevesi sunar.	Yalnızca Qiskit'te kodlanmış kuantum programlarının test edilmesini desteklemektedir.	Sınırlı sayıda ölçüm yapılmıştır.	[2]
Kapsayıcı Kriter	QsharpTester	Simülatör desteği ile fiziksel kuantum sistemine ihtiyaç duymadan çalıştırma testi yapılabilir.	Yürütme verimliliklerinin karşılaştırılabileceği yeterli sayıda çalışma henüz yapılmamıştır.	Sınırlı sayıda ölçüm yapılmıştır.	[9]
Kuantum Mutasyon Analizi	Muskit	-Kuantum kapılarına dayalı mutasyon testi sağlar. - Özelleştirilebilir mutasyon operatörleri sunar.	Şimdilik eşdeğer mutantları tespit edemiyor.	Sınırlı sayıda ölçüm yapılmıştır.	[18]
Arama Tabanlı Test	QuSBT	Genetik algoritmalar kullanarak test senaryosu üretir. Bu da test çeşitliliğini artırır.	Arama süresi asgari düzeyde olsa da, QuSBT'nin zamanının çoğunu simülatörü kullanarak QP'ler üzerinde test vakalarını yürütmeye harcadığını görülmüştür[15].	Sınırlı sayıda ölçüm yapılmıştır. [15] çalışmada, oluşturulan test takımında başarısız test vakalarının en az %50'sini bulmayı başardığı rapor edilmiştir.	[15]
Kombinasyonel Test	QuCAT	Kuantum devrelerinin hata toleransını ölçen simülasyon tabanlı testler sunar.	Yüksek simülasyon maliyeti, büyük devrelerde performans kısıtlamalarını beraberinde getirir.	Sınırlı sayıda ölçüm yapılmıştır.	[17]
Kuantum Platform Testi	QDiff	- Kuantum platformları arasında uyum kontrolü sağlar.	Yeterli sayıda çalışma henüz yapılmamıştır.	Varyantları oluşturmada etkilidir, gereksiz kuantum donanım veya	[21]

		- Farklı diller için kıyaslama yapar.		gürültülü simülatör çağrılarını %66 oranında azaltır [21].	
Kuantum Mutasyon Analizi	QMutPy	- Mutasyon testleri için esnek ve genişletilebilir. - Qiskit ile entegrasyon sağlanmıştır.	-Sadece belirli bir platform için optimize edilmiştir.	Yüksek mutasyon skoru ile etkili sonuçlar sağlar, ancak platform bağımsız çalışmaması nedeniyle sınırlıdır.	[19]
Fuzz Testi	QuanFuzz	Hata tespiti için rastgele giriş yapar.	Rastgele testler bazen zayıf hata kapsamı oluşturabilir.	Klasik test modellerine göre %20-60 daha fazla dal kapsamı elde etmektedir[14].	[14]
Metamorfik Testler	MorphQ	Hata tespitinde kullanılır.	Değişken özelliklerin gözlemi zordur.	Sınırlı sayıda ölçüm yapılmıştır.	[13]
Hata Tespiti	Qbugs	Hata tespitinde kullanılır.	Karmaşık kuantum algoritmaları için sağladığı destek sınırlıdır.	Kuantum hesaplamalarında tekrarlanabilirliği kolaylaştırmak için henüz iyi tanımlanmış bir ölçüt bulunmamaktadır.	[23]

Tablo 1., kuantum yazılım test araçlarının test senaryolarındaki performanslarını anlamak ve araçları seçerken avantajlarını ve sınırlılıklarını değerlendirmek için kullanılabilir. Her aracın kullanım alanları farklı olduğundan, ihtiyaçlara ve uygulama alanına göre seçim yapmak en iyi sonuçların elde edilmesi açısından katkı sağlayacaktır. [24] Bu çalışmada, kuantum yazılım testindeki durumu kapsamlı bir şekilde görüntülemek için, seçilen yayınlar üzerinde bir SWOT analizi hazırlanmıştır. Tablo 2.[24], kuantum yazılım testinin mevcut durumunu ve gelişim potansiyelini anlamak açısından yararlıdır

Tablo 2. Kuantum Yazılım Test ve tekniklerinin SWOT Analizi *(SWOT Analysis of Quantum Software Testing and techniques.)*

Kategori	Açıklama
Güçlü Yönler	- *Yüksek Hata Kapsamı:* Kuantum test yöntemleri, hataları tespit etme yeteneğine sahiptir. - *Güvenilirlik:* Test yöntemleri, yazılımın doğru çalışmasını sağlamak için güvenilir bir çerçeve sunar. - *Platformlar Arası Uyum:* Araçlar, farklı kuantum donanım platformları arasındaki uyumluluğu test edebilir.
Zayıf Yönler	- *Yüksek Hesaplama Maliyeti:* Testler, büyük kuantum devrelerinde çalıştırıldığında yoğun hesaplama kaynakları gerektirir. - *Olgunlaşmamış Teknoloji:* Çoğu test aracı gelişim aşamasındadır ve henüz tam anlamıyla olgunlaşmamıştır. - *Yetersiz Standartlar:* Kuantum testleri için endüstri standartlarının olmaması uygulamayı zorlaştırır.
Fırsatlar	- *Araştırma ve Geliştirme:* Kuantum test yöntemleri, akademik ve endüstriyel araştırmalara yatırım çekmektedir. - *Gelişen Platformlar:* Kuantum donanım ve simülatörlerindeki ilerlemeler, test süreçlerini iyileştirme fırsatları sunar. - *Yeni Uygulama Alanları:* Kuantum yazılımlarının finans, kimya, lojistik gibi alanlarda artan uygulama potansiyeli bulunmaktadır.
Tehditler	- *Klasik Bilgi İşlem ile Rekabet:* Kuantum yazılımın yavaş gelişimi, klasik sistemlerle rekabeti zorlaştırmaktadır. - *Güvenlik Riskleri:* Henüz tespit edilemeyen hatalar, yazılım güvenliği için tehdit oluşturabilir. - *Hızla Değişen Teknoloji:* Kuantum teknolojilerindeki hızlı değişim, mevcut test araçlarının hızla eskimesine yol açabilir.

Kuantum Test Mühendisliğinde Standartlaşma Problemi *(The Problem Of Standardızatıon In Quantum Test Engıneerıng)*

Kuantum test mühendisliği, klasik yazılım test mühendisliğine benzer prensipler içerse de, kuantum hesaplama ortamının getirdiği özel durumlar (belirsizlik, süperpozisyon ve doğrulama karmaşıklıkları vs.) nedeniyle klasik standartların doğrudan uygulanabilirliği sınırlıdır. Kuantum test mühendisliği henüz klasik yazılım test mühendisliği kadar yerleşik bir standart setine sahip değildir. Ancak, kuantum yazılım ve sistemlerinin test edilmesi için önerilen yaklaşımlar ve gelişmekte olan yöntemler bulunmaktadır. Bu bağlamda, kuantum test mühendisliği standartları; büyük ölçüde akademik araştırmalara, uygulama odaklı önerilere ve mevcut klasik test standartlarının uyarlanmasına dayanmaktadır. Tablo 3.'de Klasik test mühendisliğinde yer alan bazı standartların kuantum test mühendisliğine ne ölçüde uyarlanabileceği değerlendirilmiştir.

Tablo 3. Klasik test mühendisliği standartlarının kuantum test mühendisliğine uygulanabilirliği *(Applicability of classical test engineering standards to quantum test engineering)*

Standart	Kuantum Testine Uygulanabilirlik	Kaynaklar
ISO/IEC 29119	Kısmen uygulanabilir, dinamik test süreçlerinden test durum spesifikasyonları kuantum spesifikasyonlarına göre uyarlanabilir.	[25]
IEEE 829	Kısmen uygulanabilir, test tasarım ve vaka spesifikasyonlarının kuantum durumları (dolanıklık, süperpozisyon vs.) ve spesifikasyonları göz önüne alınarak revize edilebilir.	[26]
TMMi	Kısmen uygulanabilir, TMMI olgunluk seviyeleri, Kuantum test ortamının olgunluk seviyelerine göre tanımlanabilir.	[27]

Kuantum test mühendisliğinde, klasik test standartları çerçevesinde yeni model ve metodolojilerin geliştirme çalışmalarının yaygınlaştırılması, bu alanda uygulanabilirliğin artırılması için önem arz etmektedir. Quito, QuCAT, QSharpTester gibi test araçları; klasik yazılım test standartlarına uyumlu ve pratikte kullanılabilir olması sebebiyle standartlaşma potansiyeli taşımaktadır. Her bir test aracının odak noktası farklıdır. Tek bir standart setinin oluşturulabilmesi için kuantum yazılımının kullanılabilirliğinin artırılması, kuantum test mühendisliği alanında kullanılan araç ve metodolojilerin standartlaşmasına katkı sağlayacaktır.

ÖNERİLER VE TARTIŞMA *(SUGGESTIONS AND DISCUSSION)*

Kuantum programlarına yönelik test yöntem ve araçları, gelişmekte olan kuantum yazılım mühendisliği alanı için büyük önem taşımaktadır. Kuantum programlarının temel prensipleri (süperpozisyon, dolanıklık gibi) bu alanda klasik yazılım test yaklaşımlarını yetersiz kılmaktadır. Bu nedenle, yeni ve inovatif test yöntemleri gereklidir. Kuantum algoritmalarının hataya çok hassas olması nedeniyle, bu test araçları yazılımların doğruluğunu ve güvenilirliğini artırmada etkilidir. Özellikle metamorfik testler, karmaşık kuantum davranışlarının doğruluğunu analiz etmeye yardımcı olur. Kuantum programlarının doğruluğu hataların erken tespiti ile sağlanabilir. Kuantum test araçları, mutasyon analizi veya kombinasyonel testler ile kuantum devrelerinin doğru çalışıp çalışmadığı denetlenebilir. QDiff gibi araçlarsa, farklı kuantum platformları arasında uyum kontrolü sağlayarak, farklı kuantum işlemcilerindeki devre performansını kıyaslamaya yardımcı olur. Bu sayede platformdan bağımsız yazılım geliştirmeye olanak tanır. Quito aracı belirli kuantum test senaryolarında etkin bir test ortamı sunarken, QMutPy platform bağımlılığı nedeniyle sınırlı kalmaktadır. Kuantum mekaniksel prensipleri temel alan test yöntemleri, oldukça karmaşıktır ve doğru şekilde uygulamak için ileri düzey bilgi gerektirir. Birçok kuantum test aracı, karmaşık devrelerde veya geniş veri kümelerinde çalıştırıldığında yoğun hesaplama gücü gerektirir. Bu durum, büyük ve karmaşık algoritmaların test edilmesini zorlaştırmaktadır. QSharpTester gibi araçlar belirli diller için optimize edilmiştir. Bu, genel bir çözüm sunmak yerine sadece belirli platformlar için kullanılabilirlik sağlamaktadır. Bu çalışma, kuantum yazılım test teknikleri arasında belirgin avantaj ve dezavantajlar olduğunu göstermektedir.

Kuantum mühendisliğinde resmi standartların olmamasının temel sebebi, kuantum yazılımının çoğunlukla deneysel ve akademik ortamda geliştirilmeye devam ediyor olmasıdır. Endüstriyel kullanımın yaygınlaşması; akademik ve sektörel işbirliğinin sağlanarak küresel standartların oluşturulması için kullanım alanına özel araç ve metodolojinin geliştirilmesi gerekmektedir.

Özetle, kuantum programlarının test edilmesi, hem teorik hem de pratik açıdan birçok zorluğu beraberinde getirir. Fakat, bu çalışmada bahsedilen araç ve yöntemler, güvenilir kuantum yazılım geliştirme açısından vazgeçilmezdir. Araştırmacılar, bu yöntemleri geliştirerek testlerin verimliliğini artırmayı hedeflemektedir. Test süreçleri iyileştirildiğinde, kuantum algoritmalarının günlük hayatta daha yaygın hale geleceği ve ortak standartların oluşturulacağı öngörülmektedir. Gelecek çalışmalar, kuantum bilgisayarlarına yönelik olarak geliştirilen yazılımları, makalemizde incelediğimiz araç ve yöntemlerle sınayarak deneysel bir karşılaştırma yapmayı amaçlamaktadır.

KAYNAKÇA *(Source)*

[1] ISTQB not-for-profit association [Internet]. 2024 [a.yer 17 Aralık 2024]. Managing the Test Team. Erişim adresi: https://www.istqb.orgmanaging-the-test-team

[2] Wang X, Arcaini P, Yue T, Ali S. Quito: a Coverage-Guided Test Generator for Quantum Programs. Içinde: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) [Internet]. Melbourne, Australia:

IEEE; 2021 [a.yer 26 Aralık 2024]. s. 1237-41. Erişim adresi: https://ieeexplore.ieee.org/document/9678798/

[3] What Is Quantum Computing? | IBM [İnternet]. 2024 [a.yer 07 Şubat 2025]. Erişim adresi: https://www.ibm.com/think/topics/quantum-computing

[4] Ali S, Yue T. Quantum Software Testing: A Brief Introduction. İçinde: 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) [İnternet]. 2023 [a.yer 30 Kasım 2024]. s. 332-3. Erişim adresi: https://ieeexplore.ieee.org/document/10172759

[5] McClean JR, Romero J, Babbush R, Aspuru-Guzik A. The theory of variational hybrid quantum-classical algorithms. New J Phys. Şubat 2016;18(2):023023.

[6] Li G, Zhou L, Yu N, Ding Y, Ying M, Xie Y. Projection-based runtime assertions for testing and debugging Quantum programs. Proc ACM Program Lang. 13 Kasım 2020;4(OOPSLA):1-29.

[7] Huang Y, Martonosi M. Statistical Assertions for Validating Patterns and Finding Bugs in Quantum Programs [İnternet]. arXiv; 2019 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/1905.09721

[8] Abreu R, Fernandes JP, Llana L, Tavares G. Metamorphic testing of oracle quantum programs: 3rd IEEE/ACM International Workshop on Quantum Software Engineering, Q-SE 2022. Proc - 3rd Int Workshop Quantum Softw Eng Q-SE 2022. 2022;16-23.

[9] Long P, Zhao J. Testing Quantum Programs with Multiple Subroutines [İnternet]. arXiv; 2023 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/2208.09206

[10] Pérez-Delgado CA. A Quantum Software Modeling Language. Içinde: Serrano MA, Pérez-Castillo R, Piattini M, editörler. Quantum Software Engineering [Internet]. Cham: Springer International Publishing; 2022 [a.yer 07 Şubat 2025]. s. 103-19. Erişim adresi: https://link.springer.com/10.1007/978-3-031-05324-5_6

[11] Quantum Software Testing: A Brief Introduction [İnternet]. [a.yer 19 Nisan 2025]. Erişim adresi: https://www.simula.no/research/quantum-software-testing-brief-introduction

[12] Wang X, Arcaini P, Yue T, Ali S. Application of Combinatorial Testing to Quantum Programs. Içinde: 2021 IEEE 21st International Conference on Software Quality, Reliability and Security (QRS) [Internet]. Hainan, China: IEEE; 2021 [a.yer 30 Kasım 2024]. s. 179-88. Erişim adresi: https://ieeexplore.ieee.org/document/9724888/

[13] Paltenghi M, Pradel M. MorphQ: Metamorphic Testing of the Qiskit Quantum Computing Platform [İnternet]. arXiv; 2023 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/2206.01111

[14] Wang J, Gao M, Jiang Y, Lou J, Gao Y, Zhang D, vd. QuanFuzz: Fuzz Testing of Quantum Program [İnternet]. arXiv; 2018 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/1810.10310

[15] Wang X, Arcaini P, Yue T, Ali S. QuSBT: Search-Based Testing of Quantum Programs [İnternet]. arXiv; 2022 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/2204.08561

[16] Wang X, Yu T, Arcaini P, Yue T, Ali S. Mutation-based test generation for quantum programs with multi-objective search. Içinde: Proceedings of the Genetic and Evolutionary Computation Conference [Internet]. Boston Massachusetts: ACM; 2022 [a.yer 30 Kasım 2024]. s. 1345-53. Erişim adresi: https://dl.acm.org/doi/10.1145/3512290.3528869

[17] Wang X, Arcaini P, Yue T, Ali S. QuCAT: A Combinatorial Testing Tool for Quantum Software [İnternet]. arXiv; 2023 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/2309.00119

[18] Mendiluze E, Ali S, Arcaini P, Yue T. Muskit: A Mutation Analysis Tool for Quantum Software Testing. Içinde: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) [Internet]. Melbourne, Australia: IEEE; 2021 [a.yer 30 Kasım 2024]. s. 1266-70. Erişim adresi: https://ieeexplore.ieee.org/document/9678563/

[19] Fortunato D, CAMPOS J, ABREU R. Mutation Testing of Quantum Programs: A Case Study With Qiskit. IEEE Trans Quantum Eng. 2022;3:1-17.

[20] Fortunato D, Campos J, Abreu R. QMutPy: a mutation testing tool for Quantum algorithms and applications in Qiskit. Içinde: Proceedings of the 31st ACM SIGSOFT International Symposium on Software Testing and Analysis [Internet]. Virtual South Korea: ACM; 2022 [a.yer 13 Mart 2025]. s. 797-800. Erişim adresi: https://dl.acm.org/doi/10.1145/3533767.3543296

[21] Wang J, Zhang Q, Xu GH, Kim M. QDiff:

Differential Testing of Quantum Software Stacks. Içinde: 2021 36th IEEE/ACM International Conference on Automated Software Engineering (ASE) [Internet]. Melbourne, Australia: IEEE; 2021 [a.yer 30 Kasım 2024]. s. 692-704. Erişim adresi: https://ieeexplore.ieee.org/document/9678792/

[22] Zhao P, Zhao J, Miao Z, Lan S. Bugs4Q: A Benchmark of Real Bugs for Quantum Programs [İnternet]. arXiv; 2021 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/2108.09744

[23] Campos J, Souto A. QBugs: A Collection of Reproducible Bugs in Quantum Algorithms and a Supporting Infrastructure to Enable Controlled Quantum Software Testing and Debugging Experiments [İnternet]. arXiv; 2021 [a.yer 30 Kasım 2024]. Erişim adresi: http://arxiv.org/abs/2103.16968

[24] García De La Barrera A, García-Rodríguez De Guzmán I, Polo M, Piattini M. Quantum software testing: State of the art. J Softw Evol Process. Nisan 2023;35(4):e2419.

[25] ISO/IEC/IEEE 29119-3:2021(en), Software and systems engineering — Software testing — Part 3: Test documentation [İnternet]. 2024 [a.yer 16 Aralık 2024]. Erişim adresi: https://www.iso.org/obp/ui/en/#iso:std:iso-iec-ieee:29119:-3:ed-2:v1:en

[26] IEEE Standard for Software Test Documentation. IEEE Std 829-1998. Aralık 1998;1-64.

[27] van Veenendaal E. Produced by the TMMi Foundation.