

**BİLGİSAYAR TABANLI SİSTEMLERDE TEST
OTOMATİZASYONUNUN TASARLANMASI VE
GERÇEKLENMESİ**

**DESIGN AND IMPLEMENTATION OF TEST
AUTOMATIZATION ON COMPUTER AIDED SYSTEMS**

ALPER AVCIOĞLU

Yrd. Doç. Dr. Mehmet DEMİRER

Tez Danışmanı

Hacettepe Üniversitesi

Lisansüstü Eğitim – Öğretim ve Sınav Yönetmeliğinin

Elektrik – Elektronik Mühendisliği Anabilim Dalı İçin Öngördüğü

YÜKSEK LİSANS TEZİ

olarak hazırlanmıştır.

2015

Alper AVCIOĞLU'nun hazırladığı “**Bilgisayar Tabanlı Sistemlerde Test Otomatizasyonunun Tasarlanması ve Gerçeklenmesi**” adlı bu çalışma aşağıdaki jüri tarafından **ELEKTRİK ELEKTRONİK MÜHENDİSLİĞİ ANABİLİM DALI'nda YÜKSEK LİSANS TEZİ** olarak kabul edilmiştir.

Prof. Dr. Selçuk GEÇİM

Başkan

Yrd. Doç. Dr. Mehmet DEMİRER

Danışman

Doç. Dr. Ali Ziya ALKAR

Üye

Doç. Dr. Harun ARTUNER

Üye

Yrd. Doç. Dr. Derya ALTUNAY

Üye

Bu tez Hacettepe Üniversitesi Fen Bilimleri Enstitüsü tarafından **YÜKSEK LİSANS TEZİ** olarak onaylanmıştır.

Prof. Dr. Fatma SEVİN DÜZ
Fen Bilimleri Enstitüsü Müdürü

Her zaman yanımda olan aileme...

ETİK

Hacettepe Üniversitesi Fen Bilimleri Enstitüsü, tez yazım kurallarına uygun olarak hazırladığım bu tez çalışmada;

- tez içindeki bütün bilgi ve belgeleri akademik kurallar çerçevesinde elde ettiğimi,
- görsel, işitsel ve yazılı tüm bilgi ve sonuçları bilimsel ahlak kurallarına uygun olarak sunduğumu,
- başkalarının eserlerinden yararlanılması durumunda ilgili eserlere bilimsel normlara uygun olarak atıfta bulunduğumu,
- atıfta bulunduğum eserlerin tümünü kaynak olarak gösterdiğimi,
- kullanılan verilerde herhangi bir tahrifat yapmadığımı,
- ve bu tezin herhangi bir bölümünü bu üniversite veya başka bir üniversitede başka bir tez çalışması olarak sunmadığımı

beyan ederim.

22/06/2015

Alper AVCIOĞLU

ÖZET

BİLGİSAYAR TABANLI SİSTEMLERDE TEST OTOMATİZASYONUNUN TASARLANMASI VE GERÇEKLENMESİ

Alper AVCIOĞLU

Yüksek Lisans, Elektrik ve Elektronik Mühendisliği Bölümü

Tez Danışmanı: Yrd. Doç. Dr. Mehmet DEMİRER

Haziran 2015, 70 sayfa

Günümüz teknolojisinde; birçok modül ve birimden oluşan sistemler günden güne giderek daha karmaşık bir hal almaktadır. Bu yüzden sistem testi en büyük zorluklardan biri haline gelmiştir. Projelerde hızla artan zorlu rekabet koşulları, müşterilerin yüksek kalite beklentisi, maliyet kalemlerinin düşürülmesi, hızlı ve verimli sonuç almak gibi parametreler nedeniyle firmalar geliştirdikleri sistem/alt sistemlerin test faaliyetlerini otomatik test araçları ile gerçekleştirmeyi ve nihai ürünün kalitesini arttırmayı hedeflemektedir.

Sistem testi stratejisi manuel ya da otomatik olabilmektedir. Daha geleneksel bir yaklaşım olan manuel bir strateji ile; testçiler programı en iyi şekilde yürüteceğini düşündükleri test senaryolarını hazırlamaktadırlar. Otomatize test stratejisi ise test senaryolarını üreten bir yazılım aracına güvenerek sürecin sıkıcılığını ortadan kaldırmayı denemektedir. Çok sayıda avantajından dolayı, otomatize test stratejisi daha güvenilirdir. Bu yüzden jenerik bir otomatize test altyapısı tasarlama ve geliştirme fikri ortaya çıkmıştır.

Bu tez kapsamında, ASELSAN tarafından geliştirilen, çok sayıda platform arayüzünden oluşan bilgisayar tabanlı KULAÇ (Tek Işınlı Derinlik Ölçüm Sonarı) sisteminin sistem (yazılım/donanım) test faaliyetlerinin otomatik hale getirilmesine

yönelik genel bir test altyapısı tasarlama ve geliştirme çalışmaları anlatılmaktadır. Söz konusu otomatize test altyapısı, benzer bilgisayar tabanlı sistemlerin test faaliyetleri için kullanılabilir ve modifiye edilebilir. Bu çalışmanın sonucunda; bilgisayar tabanlı sistemlere ilişkin sistem testlerinin otomatik hale getirilmesine yönelik genel çalışmalar KULAÇ Sistemi üzerinde gerçekleştirilmiş olup, aşağıdaki işlemler başarıyla gerçekleştirilmiştir.

- Test faaliyetleri esnasında insan kaynaklı hatalar en aza indirgenmiştir
- Test adımlarınının 7/24 aralıksız koşturulması sağlanmıştır.
- Testler daha sık tekrarlanabilir hale gelmiştir.
- Test süreleri, test esnasında ihtiyaç duyulan insan gücü dolayısıyla iş yükü büyük ölçüde azalmış olup, verimlilik sağlanmıştır.

Anahtar Kelimeler: Sistem Testi, Manuel Test, Otomatik Test, Otomatize Test Altyapısı, Kullanıcı Arayüz Testi, Fiziksel Arayüz Testi

ABSTRACT

DESIGN AND IMPLEMENTATION OF TEST AUTOMATIZATION ON COMPUTER AIDED SYSTEMS

Alper AVCIOĞLU

**Master of Science, Department of Electrical and Electronics
Engineering**

Supervisor: Assoc. Prof. Dr. Mehmet DEMİRER

June 2015, 70 pages

In today's technology, systems, consisting of many modules/units, are getting more complex day by day. Because of that, system testing is becoming one of the bigger challenges. Thus, companies are aimed to perform system/subsystem testing facilities with automatized testing equipments and to enhance final product quality due to competition, high quality customer demands, cost cutting, getting quick and efficient results

System testing strategy can be manual or automated. With a manual strategy, the more traditional approach, testers prepare test suites that they think will best exercise the program. An automated testing strategy tries to remove the tediousness of the process by relying on a software tool that generates test cases. Due to many advantages, automated testing strategy is more reliable; therefore, the idea of designing and developing a generic automatized test setup arised.

In this thesis, we study the design and development of a generic test setup to automatize system (hardware/software testing) testing facilities for computer aided KULAÇ single beam echosounder system developed by ASELSAN, composed of many platform interfaces. This modular automatized test setup can be used and modified for similar computer aided system's testing facilities. As a result of this

study; general approaches regarding the automatization of system tests on computer aided systems are realized on KULAÇ (Single Beam Echosounder System). Thus, with the help of this modular automatized test setup, the followings are achieved;

- Minimized failures due to human error during testing facilities
- Running testing steps 7x24 continuously
- Reiterating test scenarios frequently
- Increased efficiency by reducing test duration and human resources.

Keywords: System Testing, Manuel Testing, Automated Testing, Automatized Test Setup, Graphical User Interface Testing, Physical Interface Testing

TEŞEKKÜR

Bu tezin oluşturulmasında yaptığı katkılardan ötürü tez danışmanım Yrd. Doç. Dr. Mehmet DEMİRER'e;

Sağlamış olduğu donanım imkanları ve diğer her türlü destek için ülkemizin güzide kuruluşu ASELSAN'a ve işyerindeki yöneticilerime;

Tez çalışmam boyunca destekleri ve yardımları ile yanımda olan değerli çalışma arkadaşlarım; Gökhan ORDU'ya, Tayfun ŞEN'e, Mehmet TÜRKUZZAN'a, Ahmet KARAKAYA'ya, Ruhi KARAV'a, Serkan NAS'a, Uğur AKTÜRK'e, Merve DİLEK'e Ali GÜMÜŞ'e ve Merve AYDOĞAN'a;

Her zaman varlıklarıyla bana güç katan değerli dostlarım; Görkem ÇAKICI'ya, Murat KIRKAĞAÇ'a, Mehmet Ali KURUT'a, Hasan KUMRU'ya, Melih AKSOY'a, Muhammed YAKIN'a ve Gökhan SİVRİCE'ye;

Bu çalışmanın tamamlanabilmesi hususunda, bilgi ve deneyimleri ile yol gösteren, hakkını asla ödeyemeyeceğim, her türlü desteği veren abim Abdurrahman AVCIOĞLU'na;

Hayatım boyunca olduğu gibi, tez aşamasında da bana moral veren ve dualarıyla her zaman yanımda olan annem Sabahat AVCIOĞLU ve babam Erol AVCIOĞLU'na;

Sonsuz teşekkürlerimi sunuyorum.

İÇİNDEKİLER

	<u>Sayfa</u>
ÖZET	i
ABSTRACT	iii
TEŞEKKÜR	v
İÇİNDEKİLER	vi
ŞEKİLLER	ix
ÇİZELGELER	xi
SİMGELER VE KISALTMALAR	xii
1. GİRİŞ	1
2. TEST LİTERATÜRÜ	3
2.1 Sistem Mühendisliği Süreci	4
2.1.1 Süreç Girdisi	5
2.1.2 Süreç Geliştirme	6
2.1.2.1 Gereksinim Analizi	6
2.1.2.2 Fonksiyonel Analizi / Tahsisi	6
2.1.2.3 Sentez	6
2.1.2.4 Gereksinim Döngüsü	7
2.1.2.5 Tasarım Döngüsü	7
2.1.2.6 Doğrulama	7
2.1.2.7 Sağlama	9
2.1.2.8 Sistem Analizi ve Kontrolü	9
2.1.3 Süreç Çıktısı	10
2.2 Sistem Geliştirme Modeli	11

2.3	Ürün Geliştirme Süreci	13
2.3.1	Fazlar	14
2.3.2	Kontrol Noktaları	14
2.3.3	Disiplinler	15
2.3.4	Yapay Olgular (Artefakt).....	16
2.4	Test Teknikleri.....	16
2.4.1	Kara Kutu Testi	20
2.4.2	Beyaz Kutu Testi.....	21
2.5	Test Metotları	22
2.5.1	Birim Testi	24
2.5.2	Entegrasyon Testi	26
2.5.3	Sistem Testi	27
2.5.4	Döngüde Donanım Testi (Hardware-In-the-Loop).....	29
2.5.5	Hata Enjeksiyon Testi	30
2.6	Otomatize Yazılım Testi	30
3.	TEST EDİLEN YAPI.....	32
3.1	Sisteme Genel Bakış.....	32
4.	TEST ALTYAPISI GELİŞTİRME ÇALIŞMALARI	34
4.1	Manuel Test Altyapısının Oluşturulması.....	34
4.1.1	Çevre Birimlerin Simüle Edilmesi	36
4.1.1.1	Seri Kanal1 Alt Sistem Arayüzünün Simüle Edilmesi.....	37
4.1.1.2	Seri Kanal2 Alt Sistem Arayüzünün Simüle Edilmesi.....	38
4.1.1.3	Sensör Alt Sistem Arayüzünün Simüle Edilmesi	39
4.2	Otomatik Test Altyapısının Oluşturulması	42
4.2.1	Fiziksel Arayüz Testlerinin Otomatize Edilmesi.....	44
4.2.1.1	USB Bağlantı Arayüzü	45
4.2.1.2	Ethernet Bağlantı Arayüzü	46

4.2.1.3	Güç Giriş Arayüzü.....	48
4.2.1.4	Güç Çıkış Arayüzü.....	50
4.2.1.5	Seri Haberleşme Bağlantı Arayüzü.....	52
4.2.2	Fonksiyonel Testlerin Otomatize Edilmesi	58
5.	SONUÇLAR.....	66
	KAYNAKLAR.....	68
	ÖZGEÇMİŞ	70

ŞEKİLLER

Sayfa

Şekil 1. Sistem Mühendisliği Süreç Şeması	5
Şekil 2. Test Bilgi Akışı	9
Şekil 3. Yazılım/Donanım Geliştirme Süreci V modeli	11
Şekil 4. Hata Eğilim Analizi: Zamanla Fark Edilen Hataların Sayısı	17
Şekil 5. Kara Kutu ve Beyaz Kutu Metotları Karşılaştırılması	18
Şekil 6. Kara Kutu ve Beyaz Kutu Metotları Blok Şeması.....	19
Şekil 7. KULAÇ Sistemi Genel Çalışma Senaryosu Gösterimi	32
Şekil 8. KULAÇ Sistemi Gösterimi.....	33
Şekil 9. KULAÇ Sistemi Ekranı (Kullanıcı Arayüz) Gösterimi	33
Şekil 10. Manuel Test Senaryosu.....	35
Şekil 11. TEY İçin Çevre Birim/Alt Sistemlerin Gösterimi	36
Şekil 12. Seri Kanal1 Alt Sistemi Arayüz Simülatörü	37
Şekil 13. Seri Kanal2 Alt Sistemi Simülatör Yazılımı Ekran Görüntüsü	38
Şekil 14. Seri Kanal2 Alt Sistemi Simülatör Yazılımı Alt Ekran Sekmesi Görüntüsü	39
Şekil 15. Sensör Alt Sistemi Blok Şeması	40
Şekil 16. EDI DSTS-4A Derinlik Ölçüm Test Seti (Düşük Frekans).....	41
Şekil 17. EDI DSTS-5A Derinlik Ölçüm Test Seti (Yüksek Frekans)	41
Şekil 18. Otomatik Test Senaryosu	43
Şekil 19. TEY Arayüz Blok Şeması	44
Şekil 20. USB Bağlantı Test Modülü Yazılımı Ekran Görüntüsü.....	45
Şekil 21. USB Bağlantı Test Sonuçları Ekran Görüntüsü	46
Şekil 22. Ethernet Bağlantı Test Modülü Yazılımı Ekran Görüntüsü	47
Şekil 23. USB Bağlantı Test Sonuçları Ekran Görüntüsü	48
Şekil 24. AC Güç Kaynağı Test Modülü Yazılımı Ekran Görüntüsü	49

Şekil 25. AC Güç Kaynağı Test Sonuçları Ekran Görüntüsü	50
Şekil 26. DC Gerilim Çıkış Test Modülü Yazılımı Ekran Görüntüsü	51
Şekil 27. DC Gerilim Çıkış Test Sonuçları Ekran Görüntüsü	52
Şekil 28. Seri Kanal Haberleşme Bağlantı Arayüzü Gösterimi	53
Şekil 29. Tuş Takımı Simülatör Yazılımı Ekran Görüntüsü.....	54
Şekil 30. Derinlik Simülatörü Yazılımı Ekran Görüntüsü.....	55
Şekil 31. Seri Kanal Loopback Testi Bağlantı Gösterimi	56
Şekil 32. Seri Kanal Bağlantı Test Modülü Yazılımı Ekran Görüntüsü (Başarılı)..	57
Şekil 33. Seri Kanal Bağlantı Test Modülü Yazılımı Ekran Görüntüsü (Başarısız)	57
Şekil 34. TEY GKA Otomatizasyonu Bağlantı Şema Gösterimi.....	60
Şekil 35. Testcomplete Test Adımı Başarılı Gösterimi	61
Şekil 36. Testcomplete Test Adımı Başarısız Gösterimi.....	61
Şekil 37. Testcomplete Test Betiği Kesit Gösterimi	62
Şekil 38. Testcomplete Test Sonuç Gösterimi.....	63
Şekil 39. Otomatize Test Altyapısı Blok Gösterimi	64
Şekil 40. Otomatize Test Altyapısı.....	65

ÇİZELGELER

Sayfa

Çizelge 1. Kara Kutu vs Beyaz Kutu Test Özellikleri Karşılaştırması	19
Çizelge 2. Fonksiyonel Testler ve Fonksiyonel Olmayan Testler	24
Çizelge 3. Otomatik Test Yürütme Araçları Karşılaştırma Çizelgesi	59
Çizelge 4. Test Altyapıları Karşılaştırma Çizelgesi	66

SİMGELER VE KISALTMALAR

Simgeler

≈ Yaklaşık Olarak Eşit

Kısaltmalar

AC	Alternatif Akım	Alternating Current
API	Uygulama Programlama Arayüzü	Application Programming Interface
CPU	Merkezi İşlem Birimi	Central Processing Unit
DC	Doğru Akım	Direct Current
DDT	Döngüde Donanım Testi	Hardware-In-Loop
DKB	Donanım Konfigürasyon Birimi	Hardware Configuration Unit
DSTS	Derinlik Ölçüm Test Seti	Depth Sounder Test Set
EDI	Elektronik Aygıtlar Firması	Electronic Devices Inc.
GKA	Grafiksel Kullanıcı Arayüzü	Graphical User Interface
GPIO	Genel Amaçlı Arayüz Veri Yolu	General Purpose Interface Bus
IEEE	Elektrik Elektronik Mühendisleri Enstitüsü	Institute of Electrical and Electronics Engineers
LAN	Yerel Alan Ağı	Local Area Network
MAN	Metropolitan Alan Ağı	Metropolitan Area Network
SMS	Sistem Mühendisliği Süreci	System Engineering Process
TEY	Test Edilen Yapı	System Under Test
TGG	Test Güdümlü Geliştirme	Test Driven Development
USB	Evrensel Seri Veri Yolu	Universal Serial Bus
ÜGS	Üretim Geliştirme Süreci	Product Development Process
YKB	Yazılım Konfigürasyon Birimi	Software Configuration Unit

1. GİRİŞ

Günümüzde birçok ülke, ileri bir teknolojiye sahip olabilmek ve rakiplerinden daha da üstün bir konuma gelebilmek için sahip oldukları teknolojiyi sürekli geliştirmek, bunu yaparken de en yeni teknolojileri kendi sistemlerine entegre etme gerekliliği duymaktadırlar.

Kritik koşullar için sistem testleri gerçekleştiren birçok şirket; yapay olguların kalitesini artırma, test fazını genişletme ve daha iyi bir test tecrübesi sağlama gibi nedenlerden dolayı test yaklaşımlarını yenilemektedirler. Farklı şirketler araştırıldığında testlerin yürütüldüğü gerçek ortamların (sistemlerin) simüle edilmiş (benzetilmiş) ortamlarla (sistemlerle) yer değiştirmesi fark edilen en yaygın düşüncedir. Bu eğilim gerçekte, geliştirme maliyetlerinin düşürülmesine, yazılım ve donanımın ayrı geliştirilmesinden dolayı geliştirme sürelerinin kısalmasına, kritik koşullarda çalışan sistemlerin sistem kalitesinin artırılmasını vaat etmektedir.

Sistem test faaliyetleri, tüm sistem geliştirme süreci boyunca süregelen, sadece hataların bulunup ayıklanması işlemlerini değil, hata oluşmasını önleyici yaklaşımların uygulanmasını da içeren faaliyetlerdir. Gereksinimlerin belirlenmesi aşamasından itibaren başlayan bu faaliyetler sistemlerin teslim aşamasına kadar farklı disiplinlere uygun olarak çalışılmasını gerektirmektedir.

Sistem testlerinin;

- Sistemin kullanıcı ihtiyaçlarını karşıladığını görmek ve müşteriye sunmadan önce ürünün kalitesinden emin olmak
- Yeniden çalışma (düzeltme) ve geliştirme masraflarını azaltmak
- Geliştirme işleminin erken aşamalarında yanlışları saptayarak ileri aşamalara yayılmasını önlemek, böylece zaman ve maliyetten tasarruf sağlamak
- Müşteri memnuniyetini arttırmak ve izleyen siparişler için zemin hazırlamak gibi nedenlerden dolayı günümüzde gerçekleştirilmesi gereklilik haline gelmektedir [1].

Bu çalışma kapsamında yapılan test altyapısı çalışmaları, derinlik ölçüm sonarı sisteminin sistem test doğrulama faaliyetlerini içermektedir. ASELSAN tarafından

geliştirilen KULAÇ Derinlik ölçüm sonarı; bir su üstü veya su altı platformuna entegre olan, dalış derinliğini ve/veya deniz dibi derinliğini ölçen sistem olup, KULAÇ Sisteminin birimleri, fiziksel arayüz ve fonksiyonel açıdan test işlemlerine tabi olmaktadır. Tasarlanan ve geliştirilen test altyapısı fiziksel arayüz ve fonksiyonel testlerin koşturulması ve belirlenen senaryolar ile test faaliyetlerinin tamamlanmasını içermektedir.

Bu tez çalışmasında ilk olarak çalışma kapsamında literatür taraması yapılmış ve çalışma ile ilgili bilgiler toplanmıştır. Daha önce konu ile ilgili oluşturulan makaleler incelenmiş ve yaklaşımlar değerlendirilmiştir.

İkinci bölümde, test literatürü için gerekli sistem mühendisliği, üretim geliştirme süreçlerinden bahsedilerek test metodolojisine yönelik yaklaşımlardan bahsedilmiştir.

Üçüncü bölümde Test Edilen Yapı (TEY) ve bu birime ait genel bilgiler verilmiş, dördüncü bölümde ise TEY'in sistem testlerinin deniz ortamından önce entegrasyon ortamında doğrulanması ve üretim sürecinde sistem testlerinin laboratuvar ortamında gerçekleştirilmesi amacıyla çevre birimlerin/alt sistemlerin simüle edilmesine yönelik çalışmalar ve manuel (elle yapılan) testlerin otomatizasyonunun sağlanabilmesi amacıyla tasarlanan otomatize test atyapısına yönelik çalışmalar belirtilmiştir.

Beşinci ve son bölümde ise manuel test yaklaşımı ve otomatize test yaklaşımı karşılaştırılmalı olarak incelenmiş ve sonuçlara yer verilmiştir. Aynı zamanda bu bölümde gelecek dönem çalışmalar için öneriler yer almaktadır.

2. TEST LİTERATÜRÜ

Günümüzde, yaşamımızı kolaylaştırmak için günlük operasyonel işleri yapmak için, gömülü aygıtlar günden güne daha sıklıkla kullanılmaktadır. Bilgisayarlar, cep telefonları, tabletler, navigasyon vb. cihazlar farklı amaçlar için geliştirildiğinden, cihazlarda kullanılan farklı programlama dilleri ve işletim sistemleri ciddi büyüklüktedir.

Cihazların büyük çoğunluğunda birçok yazılım kritik olmayan koşullarda çalışırken, diğerleri kritik koşullarda çalışmaktadır. Kritik koşullarda kullanılan yazılım/donanımda ortaya çıkabilecek hatalar, güvenlik ve maliyet açısından tehlikeli sonuçlar doğurabilmektedir.

Son zamanlarda sistem testleri, güvenlik, güvenilirlik, performans ve uygunluk açısından sistemin kalitesinin iyileştirilmesinden dolayı giderek önem kazanmaktadır. Sistem geliştirme esnasında, yazılım/donanım geliştiriciler genellikle eforlarının %50'sini test faaliyetlerine harcamaktadır. Bu yüzden geliştirme fazı maliyet ve zaman açısından önemli hale gelmektedir.

Sistem ölçü ve karmaşıklık bakımından büyüdükçe, sistemde hata alınan potansiyel noktaların sayısı da artmaktadır. Bu durumlarda test faaliyetleri de iki ana nedenden dolayı daha karmaşık hale gelmektedir:

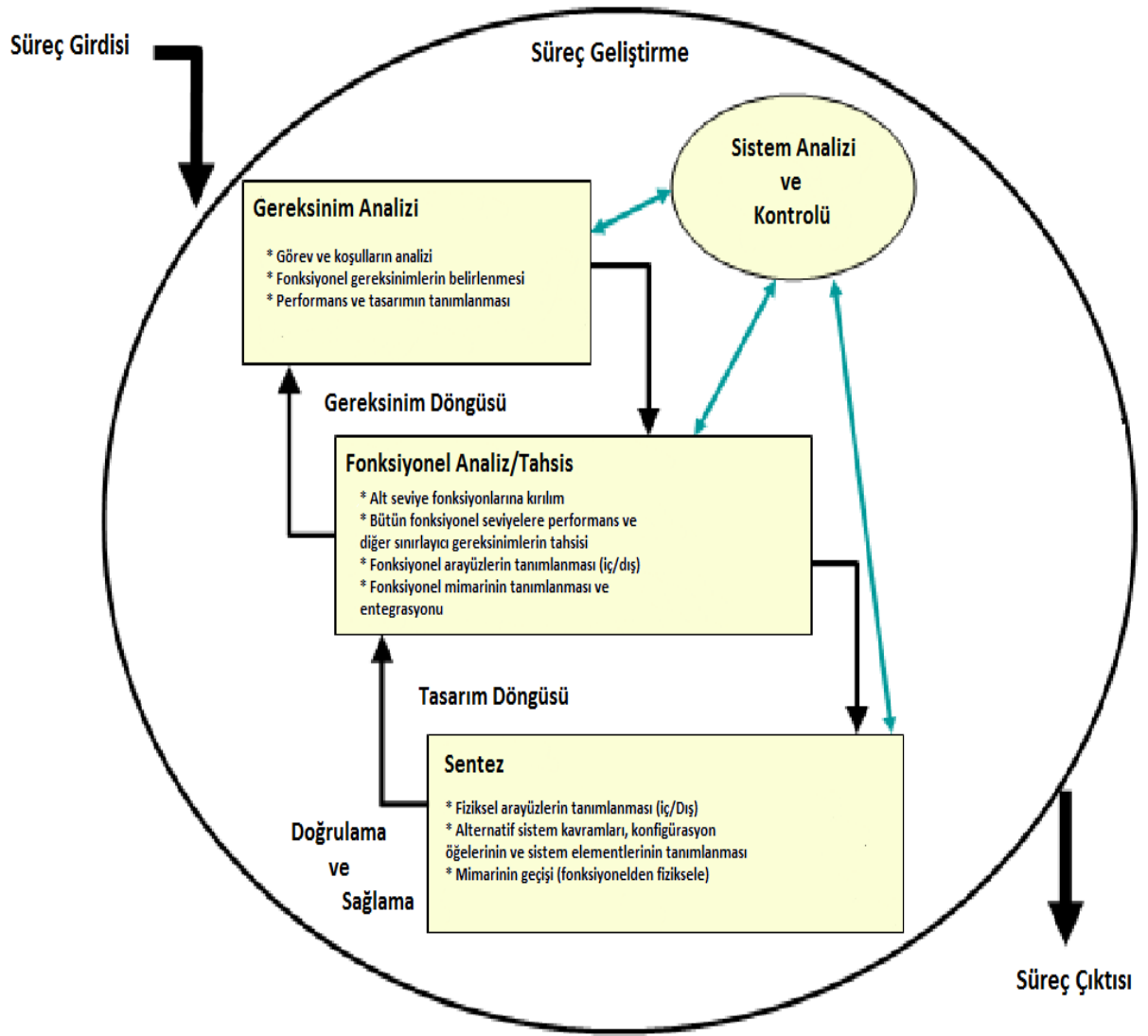
- Testin doğruluklarını kontrol etmek için mümkün olan tüm potansiyel davranış kombinasyonlarının simule edilmesi yoluyla, test işlerini organize etmek çok zordur.
- Sistemin büyük bir kısmını ve tepkilerini keşfetmek için otomatize edilmiş kapsam mekanizması işletildiği zaman bile, karşılık gelen çıktılar ve girdilerin doğruluğunu değerlendirebilen otomatize bir ortam yaratmak problem haline gelmektedir.

İlerleyen bölümlerde sistem mühendisliği süreci, sistem geliştirme modeli ve ürün geliştirme süreci ile ilgili farklı alanlarda sistem testlerinin nasıl ve neden gerçekleştirildiği ile ilgili genel bilgi verilmiştir. Ayrıca, geliştirme esnasında sisteme uygulanabilen bazı test teknik ve metotları analiz edilmiştir.

2.1 Sistem Mühendisliđi Süreci

Sistem Mühendisliđi Süreci (SMS) tüm yazılım ve donanım konfigürasyon birimi (YKB ve DKB) geliştirme faaliyetlerinin temelinde yer alan döngü ve öz yineli bir süreçtir. Sistem mühendisliđi tekniklerinin YKB ve DKB geliştirme çalışmaları esnasında uygulanması ve kullanılması kaçınılmazdır. SMS, yazılım ve donanımların oluşturulması için izlenen ve ilgili aktiviteler ile adlandırılan aşamaları temsil etmektedir [3].

Şekil 1'de görüldüğü gibi aktiviteler yukarıdan aşağıya sırayla uygulanmaktadır. SMS'de yukarıdan başlayarak süreç girdisi ilk aktiviteyi oluşturmaktadır. İkinci aktivite ise süreç geliştirme olmakla birlikte; gereksinim analizi, gereksinim döngüsü, fonksiyonel analizi/tahsisi, tasarım döngüsü, sentez, doğrulama ve sağlamadan oluşmaktadır. Üçüncü aktivite ise süreç çıktısını temsil etmektedir [2].



Şekil 1.Sistem Mühendisliği Süreç Şeması

2.1.1 Süreç Girdisi

Süreç girdisi; hedefler, müşterilerden gelen istekler ve gereksinimler gibi resmi ve resmi olmayan bilgilerin toplandığı donanım ve yazılım geliştirme süreçlerinin ilk basamağıdır. Ayrıca bu fazda görevler, fayda ölçümü, koşullar, sınırlamalar, mevcut teknoloji, önceki geliştirmeden gelen gereksinim çıktıları, program karar gereksinimleri ve standart ve teknik özellikten dolayı uygulanan gereksinimler dikkate alınmaktadır [1]. Bilgilerin toplanmasından sonra, geliştirmenin ilk bölümü olan bu bölümün sonuçları süreç geliştirme bölümüne girdi oluşturmaktadır.

2.1.2 Süreç Geliştirme

Süreç girdisinde toplanan verilerin analizinin gerçekleştirildiği süreç geliştirme aşaması, donanım ve yazılım geliştirme süreçlerinin temel kısmını oluşturmaktadır. SMS için, sonraki diğer fazlar süreç geliştirme aşamasına dayanarak planlanmaktadır [1].

2.1.2.1 Gereksinim Analizi

Bu fazda, müşteri isteklerine dayanarak toplanan bilgiler dikkatlice değerlendirilip analiz edilmektedir. Gereksinimler içindeki belirsizlikler daha anlaşılabilir, tam ve açık hale getirilerek, gereksinimlerin anlaşılabilirliği sağlanmaktadır. Tasarım ve performans kısıtlayıcı gereksinimlerin yeniden tanımlanması ve fonksiyonel gereksinimlerin belirlenmesi bu basamakta gerçekleştirilmekte olup, daha sonra bunlar, yazılım ve donanımlarda olması gereken işlevselliklerin, gerekliliklerinin toplanmasına dönüştürülür [1].

2.1.2.2 Fonksiyonel Analizi / Tahsisi

Gereksinim toplama süreci, alt seviye detaylarının keşfedilmesi için çok erken bir faz olduğundan, gereksinimlerin toplanması üst seviye fonksiyonlar grubu başlığı altında sağlanmaktadır. Bu yüzden bu basamaktaki ilk aktivite üst seviye fonksiyonların (gereksinimlerin) alt seviye fonksiyonlarına parçalanarak çözümlenmesidir. Buna ilaveten performans ve diğer kısıtlayıcı gereksinimler tüm fonksiyonel seviyelere bu basamakta dahil edilmekte olup, iç ve dış fonksiyonel arayüzlerin ve fonksiyonel mimarinin tanımlanması; tanımlanan fonksiyonel mimarinin entegrasyonu da bu bölümde gerçekleştirilmektedir [1].

2.1.2.3 Sentez

Tasarımların birleştirilmesi olarak da adlandırılan bu geliştirme fazı, fonksiyonel mimarinin fiziksel mimariye dönüşümünden sorumlu fazdır. Fiziksel mimarinin her bir kısmı, en az bir fonksiyonel gereksinimle eşleşmek zorundadır ve mimarinin herhangi bir kısmı da birçok fonksiyonu destekleyebilmektedir. Alternatif sistem kavramları, konfigürasyon öğeleri, sistem bileşenleri de bu fazda tanımlanmakta olup bu fazdan ötürü tercihli ürünler ve süreç çözümleri seçilirken, iç ve dış fiziksel arayüzler tanımlanmakta ve geliştirilmektedir [1].

2.1.2.4 Gereksinim Döngüsü

Özel bir faz olan gereksinim döngüsü, gereksinimlerin yeniden gözden geçirilmesi olarak da bilinmektedir. Gereksinim ve fonksiyonel analiz çıktıları göz önünde bulundurularak birbirleri arasında kıyaslama yapılmaktadır. Buradaki amaç gereksinim analizi ve fonksiyonel analizi/tahsisi çıktıları arasındaki tutarlılığın ve izlenebilirliğin korunmasını sağlamaktır. Buna ek olarak fonksiyonel analiz ışığında başlangıçtaki gereksinimlerin karşılanması ve yeniden değerlendirilmesinin doğruluğu da gerçekleştirilmektedir [1].

2.1.2.5 Tasarım Döngüsü

Fonksiyonel mimarinin bir ya da daha fazla ögesi ile fiziksel mimarinin her bir kısmının izlendiğini kontrol etmek için önceki geliştirme fazlarının yeniden ziyaret edilmesine izin veren süreç "tasarım döngüsü" olarak adlandırılmaktadır. Ayrıca sentez aktivitesini optimize etmek ve geliştirmek için sistemin içerdiği görevlerin yeniden kontrol edilmesine de olanak tanımaktadır [1].

2.1.2.6 Doğrulama

Sistem ya da bileşenlerin (YKB-DKB) değerlendirildiği doğrulama sürecinde, geliştirme fazındaki ürünlerin bu fazın başlangıcında oluşturulan koşulları karşılayıp karşılamadığı belirlenmektedir.

Donanım ve yazılımların doğrulaması aşaması, sistem testlerinin gerçekleştirilmesi aşaması için en önemli ve karmaşık olarak gözüken fazdır. Amacı, ilk basamakta yapılan analizden çıkan gereksinimleri ve teknik özellikleri sağlayan donanım ve yazılımların oluşturulmasını doğrulamak ve temin etmektir. Yapılan işin doğruluğundan emin olduğu doğrulama ve sağlama süreçleri geliştirme döngüsünün son basamaklarını oluşturmaktadır [1].

Bu aşamada donanım ve yazılımlar üzerinde beklenmeyen davranışlar ya da kusurları bulmak için detaylı bir test gerçekleştirilir. Genellikle, bu aşamada yazılım testine özel olarak yazılımın durumu veya istenmeyen halini göstermek için özel bir adlandırma kullanılmaktadır. İstenmeyen durumlar, tanımları ile birlikte aşağıda belirtilmiştir:

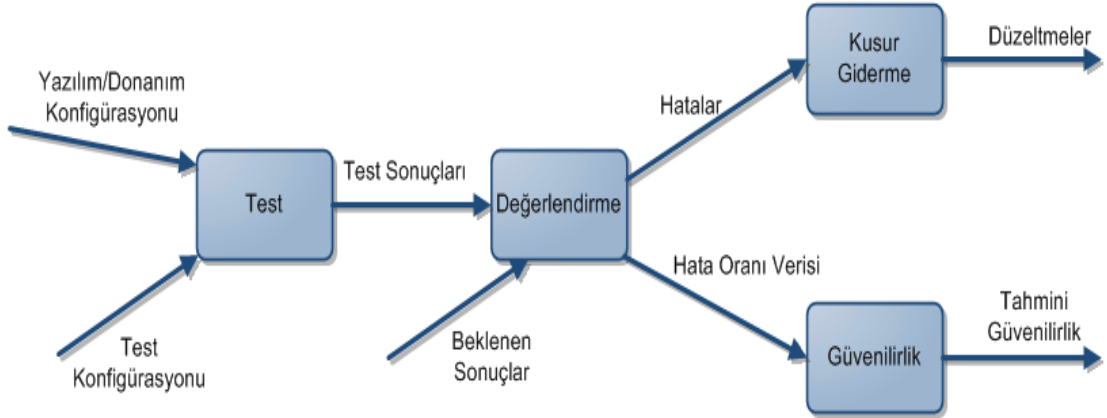
- Kod Hatası (fault): "Programın içinde yer alan yanlış bir basamak, süreç, ya da veri tanımıdır" Yazılan kod ile ilgili olup, yazılımcının kodu yazarken

yaptığı bir yanlıştan kaynaklıdır. Başka bir deyişle hata(error)'dan dolayı kodda ortaya çıkan yanlışıklardır.

- Sistem Hatası (failure): “Belirlenmiş performans gereksinimi içinde bir sistemin ya da donanım ve yazılımların gerekli fonksiyonunu yerine getirmedeki yetersizliğidir.”
- Hata (error): “ Hesaplanan, gözlenen ya da ölçülen bir değer yada koşul ile doğru, belirli yada teoratik olarak doğru değer yada koşulun arasındaki farktır”. Bir hata(fault) tarafından oluşmaktadır ve genellikle insan hatasındandır.

Bu faz sayesinde DKB ile YKB'lere ilişkin konfigürasyon birimlerinin güvenilirliği, güvenliği ve sağlamlığı ile ilgili geribildirim almak mümkün olmaktadır. Bu bakımdan sistemin doğrulanması fazı, test ve değerlendirme aşaması için gerekli olan önemli noktalardan biri olarak göze çarpmaktadır.

Test fazında yer alan tüm aktivite ve bilgi bileşenlerini Şekil 2'de yer alan test bilgi akışında gösterilmektedir. Test bilgi akış şemasına göre, test esnasında ilgili operasyonları temsil eden 4 adet düğüm noktası bulunmaktadır. Test faaliyetini temsil eden ilk düğüm olan test düğümü, testi yürütmek için gerekli olan yazılım/donanım konfigürasyonu ve test konfigürasyonunu girdi olarak alır ve test sonuçlarını çıktı olarak verir. Bu çıktı değerlendirme düğümüne girdi oluşturmaktadır. Aynı zamanda bu düğüm gereksinim analizinden gelen beklenen sonuçlar ile gerçek test sonuçlarının karşılaştırılmasıyla ilgilenmektedir. İki girdinin kıyaslanması 2 farklı çeşit çıktı vermektedir. Bunlar hatalar ve hata oranı verisidir. Değerlendirme düğümünde oluşturulan hatalar kusur giderme düğümüne girdi oluşturmakta olup, kusur giderme düğümü, test edilmekte olan sisteme düzeltmelerin uygulanmasıyla ilgilenmektedir. Son çıktı ise hata oranı verisidir ve güvenilirlik modeli düğümüne yazılım ve donanımların güvenilirliği ile ilgili bilgi sağlamak amacıyla girdi oluşturmaktadır [35].



Şekil 2. Test Bilgi Akışı

2.1.2.7 Sağlama

Sistem ya da bileşenlerin (YKB-DKB) değerlendirildiği sağlama sürecinde, geliştirme süreci esnasında ya da bitiminde belirlenmiş gereksinimlerin sağlanıp sağlanmadığı belirlenmektedir.

Gereksinim döngüsü ve tasarım döngüsüne benzer şekilde, bu fazda yazılım/donanımların davranışı ya da özelliklerinin gereksinim teknik özelliklerine uyup uymadığına bakılmaktadır. YKB açısından yazılımcılar gereksinimlerde yazan bazı özel detayları unutabileceği için bu faz büyük önem taşımaktadır. Ayrıca gereksinimler arasındaki çatışma, muğlaklık ve açıklık eksikliğinden dolayı sağlama süreci başarısız olabilmektedir.

2.1.2.8 Sistem Analizi ve Kontrolü

Sistem analizi ve kontrolü süreci tüm proje süresince yapılan bir aktivitedir. İlerleme ölçümü, alternatiflerin değerlendirilmesi ve seçimi, veri ile kararların dokümantasyonu gibi çok çeşitli durumları içermektedir. Uygun kararların alınmasına ve konfigürasyon birimlerinin daha iyi geliştirilmesine yardımcı olmasından dolayı belirtilen durumlar yazılım/donanımların geliştirme süresince büyük önem taşımaktadır.

Yukarıda belirtilen durumların değerlendirilmesi için gerekli parametreler aşağıda belirtilmiştir:

- **Ölçüp biçme çalışmaları:** Geliştirme sürecinde alternatif yaklaşımları arasında bir karar verilmesi veya seçim yapılması noktasındaki alternatiflerin nesnel olarak değerlendirilmesi aşamasıdır. Ölçüp biçme çalışmaları (Ölçüp biçme analizi) dokümantasyon sürecini belgelendiren kullanışlı bir araçtır. Bu aynı zamanda YKB üzerinde koşan yazılımların kalitesinin de değişmeden kalmasını garanti etmektedir [4] [5].
- **Risk Yönetimi:** Risklerin belirlenmesi, değerlendirilmesi ve önceliklendirilmesi aktiviteleri, geliştirme sürecini belli kabul edilebilir seviyelerde tutmaya destek olmaktadır. Ayrıca, belirtilen aktiviteler, negatif etkilerin ihtimallerinin ve etkilerinin izlenmesi, kontrolü ve azaltılması konusunda çok güçlü bir araç olarak göze çarpmaktadır. [6].

Sistem Analizi ve Kontrolü fazı; etkililik analizi, konfigürasyon yönetimi, arayüz yönetimi, veri yönetimi, olay bazlı planlama, teknik performans yönetimi ve teknik gözden geçirmeleri içeren performans tabanlı ilerleme ölçümünü verilerini içermektedir.

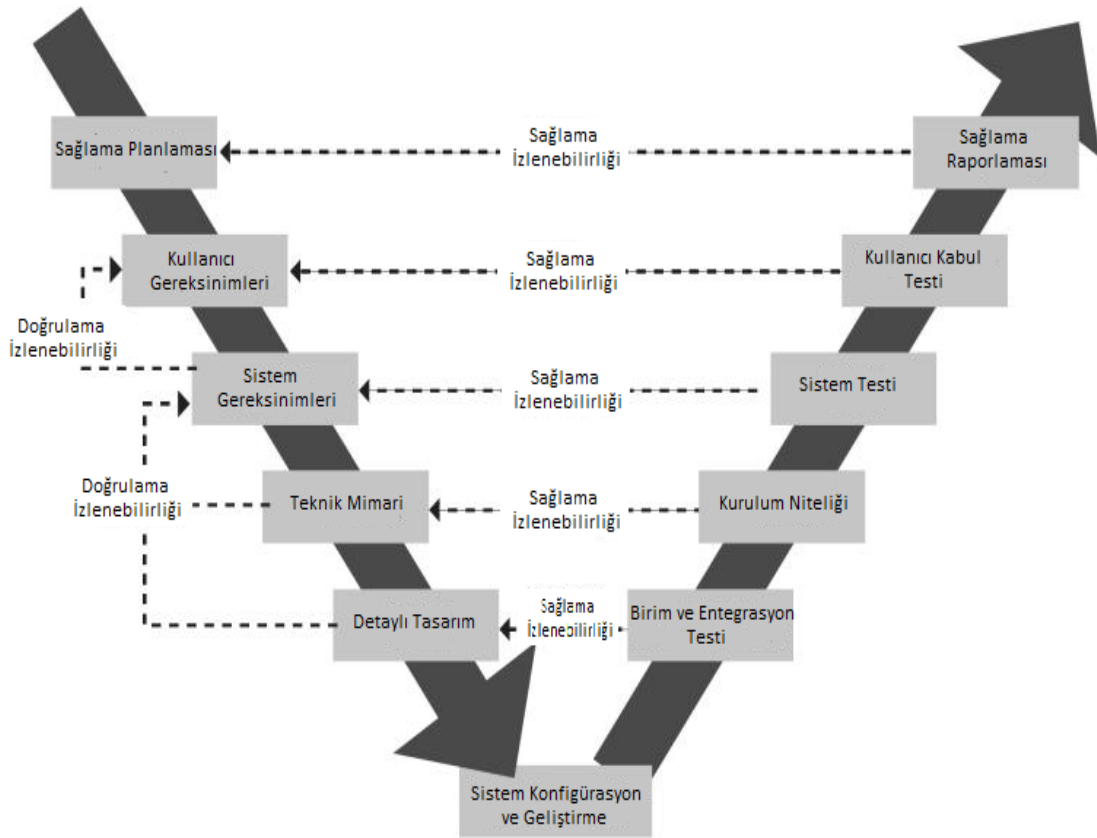
Sistem analizi ve kontrolünün amacı, projenin tüm parçalarına bakarak, geliştirme süresince kritik olan ve olmayan noktaların dikkatlice değerlendirildiğinden emin olmak ve denetlemektir.

2.1.3 Süreç Çıktısı

Süreç çıktısı, yazılım/donanım geliştirme faaliyetlerinin son fazını oluşturmaktadır. Genellikle bu basamakta, geliştirme seviyesine bağlı olarak, YKB ve DKB'lerin döngülü test tekniği yöntemiyle test edilmesiyle, birçok farklı yazılım/donanım versiyonuna (sürümüne) yönelik çıktılar oluşturulmaktadır. Bu süreçte her bir çıktı üretildiğinde, konfigürasyon birimleri için mevcut versiyona yönelik, yanlışlar (bug) ve problemler gibi geribildirimler test sorumlusu tarafından toplanır ve test sonucu şeklinde raporlanır. İlgili test sonuç raporuna istinaden konfigürasyon birimlerine yönelik hatalar/eksikler giderilerek, yeni versiyon YKB/DKB'ler için yeniden çıktı üretilmesi amacıyla test süreci tekrar başlatılmaktadır. Test sonuçları tamamen başarılı olana kadar döngülü test tekniği süreci devam etmektedir.

2.2 Sistem Geliştirme Modeli

YKB'lerin oluşturulması esnasında takip edilecek geliştirme adımlarının daha iyi anlaşılması için Şekil 1'de görülen modelin benzer özelliklerini belirten başka bir model gösterilmektedir. "V modeli" olarak bilinen Şekil 3, geleneksel şelale modelinin uzatılması ile çıkarılan bir standart sistem mühendisliği modelidir. V modeli, yazılım geliştiricilerin belirli geliştirme kriterlerini korumalarına imkan sağlayan basit bir yazılım geliştirme sürecidir. V modeli 2 ana bileşenden oluşmaktadır. Sol tarafı temsil eden birinci bileşen, yazılım geliştirme sürecini temsil ederken, sağ tarafı temsil eden ikinci bileşen yazılım test sürecini temsil etmektedir. V modeli süreci literatürde yazılımların oluşturulması olarak belirtilse de DKB'lerin oluşturulması esnasında da aynı model referans olarak kullanılmaktadır [7].



Şekil 3. Yazılım/Donanım Geliştirme Süreci V modeli

Şekil 3 gözlemlendiğinde, Şekil 1'e göre daha fazla geliştirme ve test durumlarının olduğunun farkına varılması mümkündür. Aslında V modeli; süreç girdisi, süreç

geliştirme ve süreç çıktısının ne olduğunun daha detaylı bir gösterimidir. V modelinin sol tarafından başlayarak sağlama planlaması aşamasının, kullanıcı gereksinimlerinin süreç girdisine karşılık geldiğini söylemek mümkündür. Modeller arasındaki izlenebilirliği korumak için sistem gereksinimleri basamağının gereksinim analizi kısmına, teknik mimari kısmının fonksiyonel analizi/tahsisine ve detaylı tasarımın senteze karşılık geldiğini söylemek mümkün olmaktadır. Sistem Konfigürasyon ve Geliştirme adlı V modelinin merkezi kısmında yer alan basamak önceki modeldeki süreç çıktısı ile eşleşmektedir. İlgili basamak, sistem konfigürasyonu ile yazılım/donanım geliştirme kısımları ile ilgilenmektedir. V modelinin sağ tarafı olan kısım ise sistem mühendisliği sürecinin her bir adımında gerçekleştirilen testler ile ilgilidir. Tüm test çeşitleri, sistem analizi ve kontrolü olarak adlandırılan model bileşeni ile önceki modelde özetlenmiştir. Ayrıca şekilde görülebileceği gibi her bir geliştirme süreci, kısmi yazılım/donanım çıktısının özelliklere uyduğundan emin olmak için mevcut basamağın doğrulaması ve sağlanması işlemini gerçekleştirmektedir [8].

Standardize edilmiş geliştirme süreçleri olarak düşünülen yukarıdaki modeller, teklif edilen amaçlara ulaşmak için geliştirme esnasında izlenecek basamakların tarif edilmesini sağlamaktadır. Genellikle yazılım geliştiriciler, isterlerini daha iyi yansıtan olgular oluşturmak için bu modelleri uyarlamaktadırlar. V model yazılım/donanım geliştirme modelinin avantaj ve dezavantajları aşağıda belirtilmiştir.

V modelinin avantajları:

- Doğrulama ve Sağlama planlarının ilk aşamalarda vurgulanması;
- Kullanışlı olması;
- Doğrulama ve Sağlamanın yalnızca son ürün için değil, teslim edilebilen tüm ürünlere uygulanması;
- Proje yönetimi ekiplerince kolay takip edilmesi.

V modelinin dezavantajları:

- Aynı anda gerçekleşebilecek olaylara imkan vermemesi;
- Fazlar arası tekrarlama olmaması;

- Risk çözümlene faaliyetlerini dahil etmemesi.

2.3 Ürün Geliştirme Süreci

Ürün Geliştirme Süreci (ÜGS), şirket ya da müşteri ürünlerinin yazılım/donanım geliştirmelerini desteklemek amacıyla kullandıkları sürecin genel adı olarak karşımıza çıkmaktadır. ÜGS'nin asıl amacı, tüm süreç boyunca geliştirme yanlışlarından kaçınmak ve geliştirmeyi doğru odak etrafında tutmak için yazılım/donanım geliştirme aşamasının belli prensiplere bağlı olduğundan emin olmaktır.

Bu bilgiler ışığında ÜGS'nin getirdiği faydalar aşağıda özetlenmiştir [9]:

- **Kalite güvencesi:** Tüm geliştirme süreçlerinin doğru bir şekilde yürütüldüğünden ve tamamlandığından emin olunması ile ilgilenir. Analiz, gereksinimlerin tanımlanması, kodlama, test etme, doğrulama ve sağlama gibi süreçlerin dikkate alındığını garanti eder [10].
- **Bireylerin bağımsızlığı:** Yazılım/donanım geliştiriciler arasında birbirinden bağımsız geliştirme yapılmasına imkan sunan bazı özel teknikler sağlar.
- **Proje yöneticileri desteği:** Proje yöneticilerine planlama, zaman planlama, bütçe yönetimi, kalite yönetimi, kaynak ayırma gibi konularda araç sağlar. Bu araçlar büyük projelerin yönetimi ve risklerin azaltılması için çok kullanışlıdır.
- **Şeffaf izleme:** Şeffaflığın korunup korunmadığının kontrolünden sorumludur.
- **Tahmin edilebilirlik:** Projeye ilişkili risk ve sorunların etkilerinin azaltılması ya da kaçınılması için yazılım/donanım geliştiricilerine yardım eder.
- **Yüksek seviyede yeniden kullanılabilirlik:** Yazılım/donanım geliştiricilerine projeler arasındaki uygunluğun korunması için önemli dokümanlar sağlar. Bu dokümanlar hem diğer geliştiricilerin kolay anlamasına hem de yeniden kullanıma imkan sağlar. Sağlanan dokümanlar:
 - Şablon
 - Kontrol Listesi

➤ Yönerge

- **Sürekli iyileştirmeler desteği:** Yazılım/donanım yaşam ömrü süresince yazılım/donanım geliştiricilerin desteklenmesi ile ilgilenir. Esas olarak, yazılım ve donanımların bakımı esnasında gösterilir.

2.3.1 Fazlar

ÜGS geliştirme aşamaları olarak da adlandırılan 4 fazdan oluşmaktadır. Her bir faz farklı odak alanlarına ayrılmıştır:

- **Hazırlık:** Bu faz, müşteri gereksinimleri gibi tüm bilgilerin toplandığı ve analiz edildiği projenin başlangıç kısmı olan bir fazdır. Bu faz aynı zamanda 2.1.1 ve 2.1.2 bölümlerinde belirtilen aşamaları yansıtmaktadır.
- **Tasarım:** Hazırlık fazının sonuçlarına dayanan bu faz hedefe nasıl ulaşılabileceğinin belirtilmesi ile ilgilenir.
- **Gerçekleşme:** Yazılımın kodlanmasının yürütüldüğü, projenin merkezi kısmıdır. Kodlama bir önceki fazların yönergesinin izlenmesi ile yapılır. Ayrıca bu kısım kod yazımına paralel olarak yürütülen yazılım ve donanım testlerini içermektedir.
- **Uygulama:** Modelin son parçası, yazılım ve donanımların tamamlanarak teslim edildiği yerdir. 2.1.3 bölümünde açıklanan süreç çıktısına karşılık gelmektedir.

2.3.2 Kontrol Noktaları

Yukarıda belirtildiği gibi, ÜGS yazılım ve donanım geliştirmenin 4 fazından oluşmaktadır. Her bir faz sonunda ve UGS'nin başlangıcında belirli fazdaki aktivitelerin yapılıp yapılmadığının doğrulanması için kontrol noktaları sağlanmaktadır. Bu kontrol noktaları, belirli geliştirme süresine kadar ne yapıldığının değerlendirme kriterini sağlamakta olup, toplamda 5 adet olan bu noktalar aşağıda sıralanmıştır:

- **Proje Başladı:** Gereksinim özelliklerine uygun olarak projenin yapılabilirliğinin değerlendirildiği ve doğrulandığı kontroldür.
- **Hazırlandı:** Belirleyici bir yol ile projeyi yürütme amacıyla projenin uygun tanımlanıp tanımlanmadığının değerlendirildiği bir kontroldür. Bu aşamada

hiç belirsiz proje özellikleri bulunmazken, hemen hemen projenin tüm sınırları bilinmektedir.

- **Tasarlandı:** Beklenen sonuçlar, zaman ve bütçe arasındaki zorluk düşünülerek yazılım/donanımların nasıl geliştirilmesi gerektiğini değerlendirmek amacıyla değerlendirilen projenin kontrolü aşamasıdır.
- **Gerçekleşti:** Konfigürasyon birimlerinin uygun olarak tasarlandığını ve geliştirme sonuçlarının beklenen sonuçlarla bağdaştığını doğrulamak için kullanılan kontroldür.
- **Proje Kapandı:** Geliştirilen YKB/DKB'lere ait çıktının kalitesini doğrulamak ve sağlamak için kullanılan kontroldür.

2.3.3 Disiplinler

ÜGS esnasında yapılan farklı işler disiplinler olarak tanımlanmaktadır. Tasarlanmış olan örnek bir ürün geliştirme sürecinde aşağıdaki 3 disiplin yer almaktadır:

- **Ortak Disiplinler:** Yazılım ve donanım arasındaki ortak disiplinlerdir.
- **Donanım Disiplinleri:** Donanım geliştirme ile ilgili olan disiplinlerdir.
- **Yazılım Disiplinleri:** Yazılım geliştirme ile ilgili olan disiplinlerdir.

Ortak disiplinler aşağıda özetlenmiştir.

- **Proje Yönetimi:** Belirli hedefleri başarmak için kaynakların planlanması, organize edilmesi, güvenceye alınması ve yönetilmesi disiplinidir. (zaman, malzeme ve diğer hususlar)
- **Ürün Dokümantasyonu:** Müşteriye teslim edilecek dokümantasyonların yazılması disiplinidir. (eğitim kitapçıkları, kullanıcı el kitapları, vb.)
- **Konfigürasyon ve Değişim Yönetimi:** Ürünün performansı ve gereksinimleriyle birlikte fonksiyonel-fiziksel nitelikleri arasındaki tutarlılığı koruyan, aynı zamanda bireyleri, takımları ve organizasyonları şimdiki durumundan gelecekteki bir duruma kaydıran esnekliği koruyan bir disiplindir.

- **Gereksinimler:** Gereksinim özelliklerini detaylı değerlendirilmesi ile ilgilenen disiplindir.
- **Güvenlik Yönetimi:** Projenin güvenlik seviyesi ve yönetimi ile ilgili dokümantasyon aktivitelerini içeren disiplindir.
- **Yaşam Döngüsü Yönetimi:** Tüm geliştirme fazlarının doğru biçimde yönetilmesi için kaynak ve talimatların sağlanması ile ilgilenen disiplindir.
- **Analiz ve Tasarım:** Uygun bir yazılım/donanım mimarisinin tanımlanması amacıyla gereksinimleri analiz etmek için kılavuzlar sağlayan disiplindir.
- **Uygulama (İmplementasyon):** YKB/DKB'lerin önceki analiz ve tasarımdan başlayarak uygulanması ile ilgilenen disiplindir. YKB/DKB birim testleri de bu disiplinde yer almaktadır.
- **Doğrulama:** Yazılım ve donanımları doğrulama ve sağlama amacıyla test faaliyetlerinin yürütülmesi ile ilgilenen disiplindir.

2.3.4 Yapay Olgular (Artefakt)

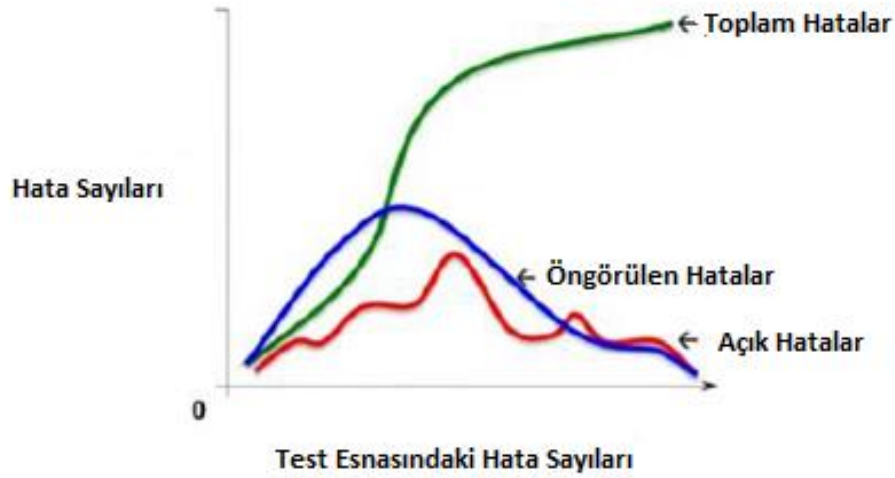
Yapay olgular geliştirme sürecinin sonunda elde edilen çıktılardır. Yukarıda anlatılan genel sistem mühendisliği sürecine dayanarak bu olgular, süreç çıktısı tarafından üretilen çıktılara karşılık gelmektedir. Test fazından sonra, yazılım ve donanımların entegrasyonu sağlanır ve dokümanlar, çizimler, tasarım modelleri ve raporlar gibi YKB/DKB'lere bağlı yapay olgular üretilir. Doğrulama ve sağlama açısından konfigürasyon birimleri olgularını mümkün olduğunca doğru yapmak için, seçilen test yöntemine karşılık gelen geliştirme süreci esnasında birçok test metodu kullanılmaktadır. Sonraki bölümde bazı büyük şirketlerde genellikle kullanılan yazılımların test edilmesine ilişkin test metodu ve teknikleri ile ilgili genel bir açıklama yer almaktadır [9].

2.4 Test Teknikleri

Sistemin olası tüm gereksinimleri sağlandığından emin olunması amacıyla verimli test senaryoları tasarlanması ve oluşturulması adımı, yazılım ve donanım testinin temel fazlarından biridir. Bu hususta, üretilen YKB ve DKB'lerin gösterdiği tüm mevcut davranış alternatiflerini kapsayabilen temsili test senaryolarını belirlemek amacıyla özel bir analiz yapılmalıdır. Ancak tipik olarak gerçekleştirilen bu analiz çalışmaları, derin bir TEY bilgisi gerektirdiğinden bu aktivite zaman kaybı olarak

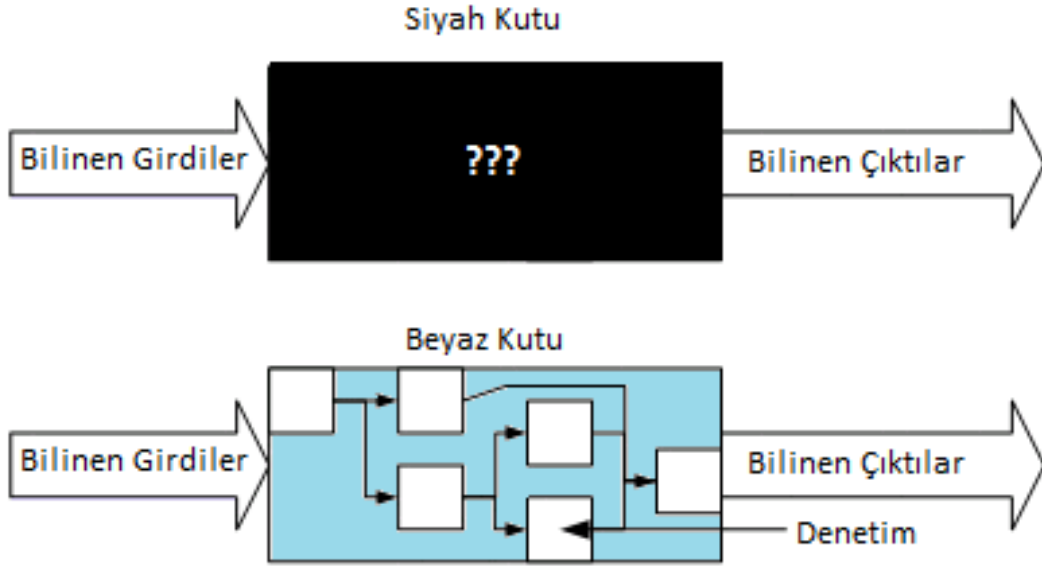
nitelendirilebilir. Yukarıda belirtildiği gibi hatasız bir yazılım/donanım sistemi elde etmek pratikte imkansızdır. Bu karmaşık sistemler için de geçerlidir. İstenen kalite seviyesine ulaşmak için mümkün olan en fazla problemi ortaya çıkarma iyi bir testin amacıdır.

Şekil 4'te görülen grafikte geliştirme ilerlemesi esnasında YKB ve DKB'lerin etkilendiği 3 farklı çeşit hata ilişkileri gösterilmektedir. Yeşil çizgi belirli bir sürede yazılım/donanım üzerinde tespit edilen hataların toplam sayısının ilerleyişini gösterirken, mavi çizgi belirli sürede öngörülen hataların sayısını göstermektedir. Mavi çizginin ilerlemesi Gaussian tipinde olup, kırmızı çizgi ise belirli süre içinde testler esnasında ortaya çıkan açık hataları göstermektedir. Kırmızı çizgi ile gösterilen açık hatalar eğrisi zaman içinde sıkça değişmesine rağmen, Gaussian ilerlemesi olarak kategorize edilmektedir [11].



Şekil 4. Hata Eğilim Analizi: Zamanla Fark Edilen Hataların Sayısı

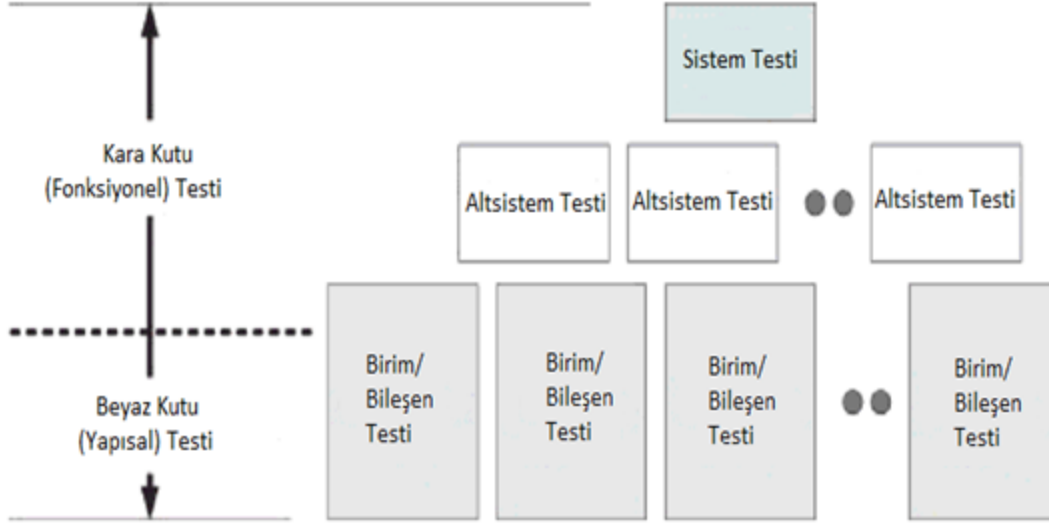
Yazılım testi, yanlış ya da hata bulunması amacıyla programların ya da uygulamaların test edilmesi sürecidir. Genelde 2 ana gruba ayrılan metotlardan oluşmakta olup; bunlar kara kutu ve beyaz kutu (saydam kutu) testi olarak adlandırılmaktadır. Test vakalarının yürütülmesinin temel amacı TEY için üretilen etkilerin analiz edilmesidir. Şekil 5'te kara kutu ve beyaz kutu metotları arasındaki karşılaştırma şekilsel olarak belirtilmiştir [12].



Şekil 5. Kara Kutu ve Beyaz Kutu Metotları Karşılaştırılması

Bu iki metot arasındaki temel fark, kara kutu testlerinin daha üst seviyede yapılır olması, diğer bir deyişle test eden kişinin sistemin detaylarını bilmesini gerektirmemesidir (uygulama kaynak kodu gibi). Beyaz kutu metodunda ise test ediciler sistemin her bir bileşeninin davranışını ve kaynak kodunu bilmek zorundadır. Genellikle, beyaz kutu testi her bir tekil yazılım bileşeninde gerçekleşmesi nedeniyle daha doğru ve daha tam görünmektedir.

Kaynak kodunu kullandığı için geliştirme süreci esnasında kullanılabilen tek test tipi olması beyaz kutu testinin avantajıdır. Kara kutu testine göre dezavantajı ise son yazılımda (sistem) kullanılmıyor olmasıdır. Şekil 6'da hiyerarşinin belirli katmanında hangi metodun kullanılacağı ve sistem testinin yazılım mimarisine uygun olarak nasıl parçalandığı görülmektedir [12].



Şekil 6. Kara Kutu ve Beyaz Kutu Metotları Blok Şeması

Çizelge 1’de kara kutu ve beyaz kutu özellikleri arasında farklı bir kıyaslama belirtilmiştir. Bu kıyaslamada göze çarpan farklılıklar; kara kutuda belirsizliğin daha çok hakim olması ve zaman maliyetinin düşük olması iken, beyaz kutunun daha belirgin ortamlarda uygulanması ve zaman maliyetinin daha yüksek olması olarak göze çarpmaktadır [13].

Çizelge 1. Kara Kutu vs Beyaz Kutu Test Özellikleri Karşılaştırması

SIRA	KARA KUTU	BE YAZ KUTU
1	Uygulamanın iç dinamikleri test uzmanı tarafından bilinmemektedir.	Test uzmanı uygulamanın tüm iç dinamiklerini bilmektedir.
2	Test sorumlusunun beklentilerine göre yapılmaktadır.	Tüm iç dinamikleri bilindiğinden uygun test datası hazırlanarak yapılmaktadır.
3	Zaman maliyeti en düşük olan test tekniğidir.	Zaman maliyeti en yüksek olan test tekniğidir.
4	Algoritma testi için uygun değildir.	Algoritma testi için uygundur.
5	Veri kısıtları bilinmediğinden, testler deneme yanılma yöntemi ile yapılır. Hatalar rastlantısal olarak tespit edilir.	Uygulamanın işlediği veri türleri ve uygulamanın ilgili veriyi işleme şekli bilindiğinden, testler bilinçli olarak uygun test datası hazırlanarak yapılır.

2.4.1 Kara Kutu Testi

Kara kutu testleri (fonksiyonel test) bir sistemin ya da bileşenin iç mekanizmasını göz ardı ederek, sadece seçilen girdi ve yürütme koşullarına karşılık üretilen çıktılara odaklanır [1].

Kara kutu testlerinin yürütülmesi (beyaz kutu ile aynı biçimde) fonksiyonel gereksinimler ya da TEY tasarım özellikleri esasında seçilen test senaryoları aracılığıyla gerçekleşmektedir.

Tipik bir kara kutu test tasarım metodu aşağıdakileri içermektedir:

- **Karar tablosu testi:** Özel koşullara dayanarak belirli faaliyetlerin yürütülmesi esnasında sistemin üstlendiği mümkün olan tüm alternatif durumların gösterildiği tabloya dayalı bir tekniktir.
- **İkili test:** Kombinasyonel sistem testini sağlamak amacıyla bir algoritmaya bağlı tekniktir. Bir çift değeri girdi olarak alır ve bu parametrelerin bütün ayırık kombinasyonlarını test eder. Eğer algoritma her bir sistem fonksiyonelliği için kullanılırsa, doğrulama test sonuçları ayrıntılı ve hızlı elde edilir.
- **Durum geçişi tabloları:** Durum geçiş tablolarına dayanan bir tekniktir. Bu tablolar durum ve olayları göstermektedir. Olaylar durumlar arasındaki geçişleri simgelerken, durumlar sistemin yürütülmesindeki tam pozisyonu simgelemektedir (mevcut durum ve sonraki durum gibi).
- **Denklik paylarına ayırma:** Bir grup değeri girdi olarak alarak, girdileri geçerli ya da geçersiz paya bölen bir tekniktir. Bu değerler sistem tarafından üretilen çıktının beklenen değerlere uygun olup olmadığını doğrulamak için kullanılmaktadır.
- **Sınır değer analizi:** Alt sistem/parça eşdeğerliğine dayanmaktadır. Gerçekte bir grup değeri alır, geçerli ya da geçersiz parçalara böler. Önceki teknikten farklı olarak, bu senaryoda sadece her bir parçanın minimum ve maksimum değerleri test edilir. Bu teknik, eşdeğer pay hakkında performans ve test eksiksizliği açısından daha etkindir.

2.4.2 Beyaz Kutu Testi

Beyaz kutu testi (yapısal test ve saydam kutu testi) bir sistemin ya da bileşenin iç mekanizmasını dikkate alan bir testtir.

Yukarıda belirtildiği ve Şekil 5 ve Şekil 6'da görüldüğü üzere beyaz kutu testi her bir konfigürasyon birimi ögesinin doğrulanmasına dayanmaktadır. Bu durum her bir tekil sistem bileşeninin testi için test senaryosunun kurgusuz olarak planlanması anlamına gelmektedir. Test senaryoları, girdileri seçen ve çıktının doğruluğunu belirlemek amacıyla test sorumluları tarafından yürütülmektedir.

Beyaz kutu testi metodu aşağıdaki bölümlerden oluşmaktadır [13]:

- **Temel yol testi:** Programcıların, test için doğrusal bağımsız yolların belirlenmesi amacıyla kullanılan bir metottur. Bu sayede yürütme esnasında mümkün olan en fazla seçenekteki test yollarını içerecek şekilde sistemin test edilmesi sağlanır. Bu metot yazılımın akış grafiğini birleştirmek için kaynak kodundan yararlanır.
- **Kontrol yapı testi:** Kontrol yapı testi aşaması 3 adet yazılım test bileşeninden meydana gelmektedir:
 - **Koşul testi:** Kaynak kodunu analiz eden ve yazılım yürütmesinde varsayılan tüm koşulları dikkat alan bir metottur.
 - **Veri akış testi:** Yazılımdaki tanımların ve değişkenlerin kullanımına göre test senaryolarının seçildiği bir metottur.
 - **Döngü testi:** Döngüleri temsil eden bir yığın özel kodu test eden bir metottur. Genel algoritma döngüler içerdiğinden ve birçok bug bunların içinde tespit edildiğinden çok kullanışlıdır. Döngüler; basit döngüler, bağlanmış döngüler, birbirinin içine yerleştirilmiş döngüler ve yapısız döngüler gibi 4 sınıfta tanımlanmaktadır.

Programlama tekniği testi: Performans testi olarak da bilinen bir metottur. Sistemin operasyonel çalışması esnasında sistemin performansını ölçmektedir. Ayrıca bu metod vasıtasıyla hafıza, CPU, disk, ağ gibi işletim sistemin seviyesindeki kaynak kullanımları izlenmektedir.

Mutasyon testi: Yazılım/donanımların değişimi/değerlendirmesi olarak tanımlanmaktadır. Bu metod ile YKB ve DKB'lere ilişkin gerçekleştirilen

modifikasyonlara ilişkin test senaryolarının, sistemin test edilebilirliği açısından yeterli olduğunun kontrol edilmesi amaçlanmaktadır. Ayrıca, önceki test sonuçlarının güvenilirliğinin bir göstergesi olarak da kullanılmaktadır.

2.5 Test Metotları

Test, yazılım/donanım yaşam döngüsünün her bir aşamasının bir parçasıdır. Ayrıca farklı içerik ve amaçlara dayanarak yazılım ve donanımlara ilişkin geliştirme aşamalarının her bir seviyesinde testler gerçekleştirilmektedir. Bugünlerde test teknikleri API'ler (Uygulama Programlama Arayüzü), kütüphaneler vb. kullanımlarını maksimize etmeye çalışan modülerize yaklaşımlar etrafında gelişmektedir. Bileşen olarak adlandırılan yeniden kullanılabilir yazılım/donanım modülleri daha iyi tasarlanmış ve test edilmiş olma avantajına sahiptir. Karmaşık bir sistem, tipik olarak eşleştiği alt problemlerle ilişkili daha küçük alt sistemlere parçalanabilmektedir. Bu sayede sistemlerin entegre edilmeden küçük alt sistemlerin kendi başına doğrulanmış olması, testler açısından güvenilir entegrasyon aşaması ve son test faaliyetlerinin gerçekleştirilmesine olanak sağlamaktadır.

Bu geliştirme yaklaşımını daha iyi anlamak için, bir puzzle yapmayı düşünebiliriz. Alt sistem bileşenlerini temsil eden birçok küçük parçalardan başlayarak, bazı parçaların birleşiminden oluşan ve büyük resmin bir bölümünü oluşturan yapıyı alt puzzle olarak adlandırabiliriz. Genellikle, puzzle boyutu büyük ise, birçok alt puzzle'ı bir araya getirmek daha uygun olacaktır. Bu alt puzzlelar altsistemleri temsil etmektedir. Bunların birleşimi bütün puzzle'ı tamamlar yani diğer bir deyişle, tüm sistemi oluşturmaktadır.

O'da sistemin nasıl parçalandığı daha iyi anlaşılabilen olup, bu yaklaşım sayesinde hem programlama hem de test bakış açılarından çok fazla kazanç elde edilebilmektedir. Elde edilen kazançlar çoğunlukla maliyet kaynaklı zaman ve para tasarruflarıdır. Bahsedilen kazançlar aşağıdaki şekilde özetlenmektedir:

- **Yeniden kullanılabilirlik:** Hâlihazırda var olan bileşenlerin kullanımının mümkün olması, bu sayede daha hızlı bir teslimat yapılması;
- **Dayanıklılık:** Tekil olarak analiz edilebilmesinden dolayı yazılım özelliklerinin yönetiminin basitleşmesi;

- **Verimlilik:** İleri derecede test edilen API'ler (Uygulama Programlama Arayüzü), kütüphaneler ya da kişisel bileşenler gibi standart bileşenlerin kullanım optimizasyonlarından dolayı yazılım yürütmenin sistem test performansını artırması;
- **Güvenilirlik:** Var olan yazılım/donanım bileşenlerinin kullanımının hata olasılığını ciddi bir şekilde düşürebilmesi, böylece sistemin güvenilirliğini artırması;
- **Taşınabilirlik:** Alt sistem bileşenleri basit operasyonları yürüten YKB ve DKB'lerden oluştuğu için, başka bir bileşeni etkilemeden yeni bir platform için hızlıca ve kolayca yeniden oluşturulabilmeleri.

Test tarafında, geliştirme esnasında daha güvenilir ve uyumlu bir test senaryosu oluşturmak için geliştirici ve testçilerin seveceği ana kazançlar, özellikle yazılım ve donanımların adım adım test edilebilme olasılığıdır. Gerçekte, test aşaması basit yazılım ve donanım bileşenlerinden başlayarak, birim testi adlı test aşamasında test edilebilmektedir. Birleştirilen bileşenler ise genellikle entegrasyon testi ile test edilebilmektedirler. Tüm sistemin doğrulandığı son test tekniği ise sistem testidir.

Sistem, birim ve entegrasyon testleri, birçok farklı test tekniklerini içeren test çeşitleri olup; bu testler literatürde fonksiyonel testler olarak nitelendirilmektedir.

- **Fonksiyonel test:** bir sistem ya da bileşenin fonksiyonel özelliklerinin doğrulanmasına/analizine dayanan bir testtir.
- **Fonksiyonel olmayan test:** Sistemin ve veya alt bileşenlerin fonksiyonel özellikleriyle ilgili olmayan bir sistem ya da bileşenin özelliklerinin testidir.

Yazılım/donanım geliştirme esnasında kullanılan bazı fonksiyonel ve fonksiyonel olmayan test teknikleri Çizelge 2'de gösterilmiştir [9].

Çizelge 2. Fonksiyonel Testler ve Fonksiyonel Olmayan Testler

Fonksiyonel Testler	Fonksiyonel Olmayan Testler
Birim Testi	Kusur Enjeksiyonu Testi
Entegrasyon Testi	Uygunluk Testi
Sistem Testi	Performans Testi
Döngüde Donanım Testi	Taşınabilirlik Testi
Onay/Kabul Testi	Ölçeklenebilirlik Testi
Regresyon Testi	Kullanılabilirlik Testi
Güvenilirlik Testi	Güvenlik Testi
Tekrar Testi	Hacim Testi
Kurgusuz Test	Stres Testi
Duman Testi	Yük Testi

Bu çalışmanın içeriğine yönelik olarak bu tekniklerden ilgili olanlar aşağıdaki bölümlerde detaylı olarak belirtilmiştir.

2.5.1 Birim Testi

Yazılım ve donanım geliştirme sürecinde kaynak kodlaması, donanıma ilişkin tasarım faaliyetlerinin gerçekleştirilmesi ve ilgili kaynakların doğrulaması işlemi birbirine paralel olarak yürütülmektedir. Birim testinin amacı hataların varlığını tespit etmek için kaynak kodu metotları ve tüm fonksiyonlar için kurgusuz test senaryoları oluşturmaktır.

Test fazı esnasında, test senaryoları başarısız olduğunda, konfigürasyon birimi sorumluları benzer şekilde YKB ve DKB'ler için yeniden tasarım sürecini işletirler.

YKB ve DKB'lerin yeniden tasarımı, konfigürasyon birimlerinin dış davranışını değiştirmeksizin iç yapısını değiştirmek ve mevcut tasarımı yeniden yapılandırmak için kullanılan bir tekniktir [14].

Yeniden tasarım süreci test prosedürüne uygun olarak, kaynak kodunda hiç hata olmadığını teyit etsin diye tüm test senaryoları başarılı bir biçimde tamamlanıncaya kadar devam eder. Beklendiği gibi testlerin başarılı olarak tamamlanması yazılımların kaynak kodunun ve donanımların hatasız olduğunu garanti etmemektedir.

Önceki etkileşimlerde başarılı ya da başarısız olan bütün test senaryolarının yeniden koşturma fazı regresyon testi olarak bilinmektedir.

Regresyon testi; yazılım/donanım geliştirme ve bakımı esnasında yapılan değişiklik ve düzeltmeler tarafından meydana gelen hataları tespit etmek amacıyla

daha önce doğru şekilde yürütülen test senaryolarının yeniden kořturulmasıdır [15].

Regresyon testi sürecinde donanımlara yönelik testlerde önemli deęişiklikler beklenmedięi için yazılımlara yönelik deęişiklikler regresyon testleri açısından önemli olarak adlandırılmaktadır. Yazılım regresyon testleri için, test senaryolarının oluşturulması 2 farklı yaklaşım aracılığı ile gerçekleştirilir. İlk olanda programcılar test senaryolarını kodlamadan sonra yazmak sorundadırlar, yazılım gereksinimleri özelliklerine dayanan “Test Gúdümlü Geliřtirme” (TGG) olarak da adlandırılan dięer yaklaşımda ise sonucun başarılı olarak düşünülmesi için sistemin testlerden geçmesi gerekmektedir. Sonuç olarak, TGG yaklaşımı aracılığıyla kaynak kodu, test senaryolarının oluşturulmasına temel olarak kullanılmamaktadır. TGG aşamasında kod yazımı, test senaryolarının yazılmasından sonra olduęu için garip görünse bile bazı önemli avantajları da beraberinde getirmekte olup, belli başlı avantajlar ařaęıda belirtilmiřtir [16].

- Doğrudan gereksinimlere dayanan test senaryoları yazmak;
- Yeniden yapılan işlerin süresini kısaltmak;
- Yazılmış olan kaynak koduna hızlı geribildirim yapmak;
- Kodları yazmadan önce bazı geliştirme uygunsuzlukları ile ilgili geribildirim vermek;
- Yazılımın deęerlendirilmesi esnasında kodu yeniden daha hızlı test etmek;
- Kalitenin arttırılması;
- Hataların azaltılması;

İlk bakıřta, TGG yaklaşımı geleneksel yaklaşıma çok benzemektedir. Doğrusu, onları birbirinden ayıran tek özellik test senaryolarının yazıldıęı zamandır.

Test Senaryosu

Bir test senaryosu, özel bir amaç için geliştirilen (belirli bir program yolu uygulamak veya özel bir gereksinimle uygunluęun doęrulanması gibi) bir grup test girdisi, yürütme kořulları ve beklenen sonuçların olduęu bir gruptur. Test senaryoları, sistemin gereksinimler tarafından belirlenen hedef deęerlerle geçmesini ya da kalmasını doęrulamak için bir yol göstermektedir.

Test senaryolarının oluşturulması aşaması testlerin gerçekleştirilebilirliği ve müşteri ile kullanıcılar açısından önemli avantajlar kazandırmakta olup, belli başlı avantajlar aşağıda belirtilmiştir.

- Test senaryolarının yürütülmesi ve dokümantasyonu geliştirme fazında hata bulma,
- Ortaya çıkarılabilir hata sayısını maksimize etme,
- Olgunlaşmayan ürün lansmanlarını önleme,
- Teknik destek maliyetleri azaltma,
- Özelliklere uygunluğu değerlendirme,
- Kurallara uyma,
- Güvenlikle ilgili riskleri azaltma,
- Hatalara rağmen ürünün kullanımı için güvenli senaryoları bulma,
- Kaliteyi değerlendirme-temin etme
- Ürünün hatasızlığını doğrulama

2.5.2 Entegrasyon Testi

Çizelge 2'ye göre entegrasyon testi fonksiyonel testlerin ikinci basamağında yer almaktadır. Birim testinin mantıksal bir sonucu olarak değerlendirilmektedir ve birim testi ile sistem testi arasındaki bir geçiş fazı olarak düşünülmektedir. Genellikle bu tip bir test 2 ya da daha fazla birimin daha önce test edildiği, bileşen olarak birleştirildiği ve aralarındaki arayüzlerin de test edildiği alt sistemlerde yürütülmektedir.

Bu durum her bir tekil birim iyi test edilmiş olsa bile, tekil birim testlerinde görülmeyen ancak bileşenlerin entegrasyonu esnasında ortaya sorunlar çıkabileceği anlamına gelmektedir.

Entegrasyon testi; tüm sistem entegre edilinceye kadar yazılım ve donanım öğelerinin etkileşimlerini değerlendirmek için gerçekleştirilen testtir [17].

Entegrasyon testi, testi yürütmek için birçok farklı yol sunar. Aşağıda entegrasyon testine yönelik en popüler bazı ortak yöntemler belirtilmektedir:

- **Big Bang:** Sonuçları test etmeden önce çoğu bileşenlerin kendileriyle bütün sistem ya da büyük alt sistemler arasındaki entegrasyonunu düzene koyan

bir yaklaşımdır. Bu tip bir yaklaşım, zaman kazanmak için çok verimlidir. Fakat hataların tespiti durumunda hataların düzeltilmesi için hızlı ve kolay bir metot sağlamamaktadır. Big bang test metodunun bir tipi kullanım modeli testi olup, bu test yazılım ve donanım entegrasyon testi aşamalarında kullanılmaktadır.

- **Yukarıdan aşağıya:** Üst seviyede entegre olan bileşenlerin ilk olarak test ve entegre edilmesini içeren yaklaşımdır. test süreci adım adım bulunduğu en aşağı seviye ulaşıncaya kadar devam eder.
- **Aşağıdan yukarıya:** Bu yaklaşımda da test süreci öncekine benzerdir. Fakat bu yaklaşımda senaryo önce basit bileşenlerden (aşağı) başlamakta ve adım adım yukarıya doğru tüm bileşenlerin entegrasyonuna kadar devam eder.
- **Sandviç:** Bu yaklaşım şemsiye adı ile de bilinmektedir. Bu yaklaşımın amacı yukarıdan aşağıya ve aşağıdan yukarıya entegrasyon testlerinden gelen avantajları birleştirmektir.

2.5.3 Sistem Testi

Sistem testi, hiyerarşik skalanın en üst seviyesinde yer almaktadır. Bu durum, sistem testinin ne alt seviye tasarımda ne de üst seviye tasarımda çalışmadığı anlamına gelmektedir. Bu tip test, yazılım ve donanımların fonksiyonel ve teknik gereksinimleri sağlayıp sağlamadığını doğrulamak amacıyla sadece fonksiyonel ve teknik gereksinimlerle birlikte çalıştırılmaktadır. Sistem testi yazılım ve donanım geliştirme aşamasının son basamaklarından biri olarak düşünülebilir. Nitekim temsili bir ortamda ürünün bir bütün olarak sağlanmasının son basamağı sistem testi aracılığı ile gerçekleştirilmektedir. Bu fazda alt sistem içindeki bileşenler ya da alt sistemler arasındaki entegrasyondan dolayı hataları tespit etmek mümkün olmaktadır.

Sistem testinin amacı, teslimat öncesinde bütün sistemin test edilmesi ve su üstüne çıkan hataları tespit etmektir.

Sistem testi kapsamında tespit edilen hatalar bireysel bileşenlere ya da iki bileşen arasındaki etkileşime dayandırılmamaktadır. Sistem testi; performans testi, güvenlik, konfigürasyon hassasiyeti, kurulum ve arıza durumundan kurtarmayı içermektedir.

Sistem testi; testçilere, uygulama mimarisi ve aynı zamanda gereksinimleri doğrulamaya ve sağlamaya imkan tanıyan kalite yönetim sürecinde kritik bir basamak olarak düşünülmektedir. Programcıların test yürüttükleri birim ve entegrasyon testinden farklı olarak, bu testte işler test uzmanları tarafından tamamlanmaktadır. Test fazı, kara kutu testi kapsamıyla yapılır, bu yüzden testçiler için derin bir kaynak kodu ya da test uygulaması için mantık bilgilerine gerek yoktur.

Başarılı testin parçaları olan ilgili faktörler aşağıda belirtilmiştir [18]:

- **Test kapsamı:** Yazılım ve donanımların ne kadar testler ile denendiğini değerlendirmek için yapılan bir ölçümdür. Kod kapsamı, vasıf kapsamı, senaryo kapsamı, ekran ögesi kapsamı ve model kapsamı gibi birçok farklı çeşit aktivitelerin gösterildiği kategoriye test kapsamı denmektedir.
- **Hata izleme:** Test senaryolarının yürütülmesi esnasında sistemde bulunan tüm hataların izinin sürülmesidir. Bu hatalar düzeltildikten sonra, sonraki test senaryolarının yürütülmesi ile önceki hataların artık olmadığı doğrulanır.
- **Test yürütme:** Testteki gizlilik seviyesini geliştirmek için test senaryolarının doğru yürütülmesini sağlayan önemli bir faktördür.
- **Yapı süreç otomasyonu:** Test fazı esnasında birçok hata tespit edildiğinden; bu sürecin otomatize edilmesi yanlış-olumlu ve yanlış-olumsuz tespitlerden kaçınılmasına, risklerin azaltılmasına, kalitenin artırılmasına ve hata düzeltilmesinin hızlandırılmasına yardım eder.
- **Test otomasyonu:** Test senaryolarının yürütülmesinin otomasyonu için çok etkili bir süreçtir. Yürütme esnasında insan hatasından kaçınarak test kalitesini iyileştirmek ve tüm testi çoğu kez yeniden yürütmek gibi avantajlar sunar.
- **Dokümantasyon:** Testçiler yürüttükleri tüm operasyonların izini sürmek zorundadır. Hata durumunda detaylı bir rapor yazılmalı ve programcılara teslim edilmelidir. Bu raporda belirtilen bilgi, hata çeşitleri, önceki koşullar, mevcut koşullar, gerçekleştirilen basamakların listesi, vb... ile ilgilidir. Genellikle hataların mümkün olan en kısa zamanda açığa çıkarılarak

programcılara yardım etmek için raporlar yazılır ve bu sayede rapor içeriğine karşılık gelen yazılım kısmıyla kolaylıkla eşleşmelidir.

Sistem testi, bir sistemde yürütülen farklı kapsamlardaki testlerin toplamı olarak düşünülmektedir. Tüm sistem, geliştirme sürecine her zaman katı bir biçimde bağlı olmaya devam etse bile farklı kapsamlardaki testler aracılığıyla tüm sistemin doğrulanması mümkün olmaktadır. Geliştirilen sistemi gerçek ortamda test etmek için olan diğer bir özel test tekniği ise donanımlı döngüdür.

2.5.4 Döngüde Donanım Testi (Hardware-In-the-Loop)

Önceki kısımda geliştirme süreci esnasında genellikle yazılım testi için kodlama ile düzeltilecek test aşamalarından bahsedilmiş ve konuya ilişkin bazı teknikler belirtilmişti. Eğer doğru yapılırsa, bu testler geliştirme altında sistemin doğruluğu hakkında güvenilir sonuçlar verebilmektedir. Ancak, birim ve entegrasyon testleri kodlama aktiviteleri ile gerçekleştirildiğinde yazılımın konuşlandırıldığı gerçek ortamdaki soyutlanmaktadır.

Döngüde donanım testi (DDT), geliştirme süreci esnasında test senaryolarını yürütmek için gerçek koşulları simüle edebilen bir testtir. Bu sayede testçilerin sistemi tam olarak test etmesine imkan tanınır. Bu tekniğin adından da anlaşılacağı gibi sistemin doğrulanması ve sağlanması, yazılımın çalıştığı gerçek ortama sanal olarak benzeyebilen bir donanıma dayanmaktadır. Elektrik Kontrol Birimi (EKB) olarak adlandırılan ilave bir gerçek donanım bileşeninden dolayı saf gerçek zaman simülasyonundan ayrılmaktadır [19].

DDT testinin getirdiği avantajlar çok sayıdadır. En önemlileri maliyet tasarrufu, zaman tasarrufu, güvenlik ve yapılabirliktir. Zaman tasarrufu ve güvenlik genel olarak maliyet ölçümü ile ilgilidir. Üstüne üstlük, testlerin ve geliştirmenin süresi planlı ürün için pazara çıkış süresini etkilemektedir. Güvenlik bakış açısı hakkında, simüle edilmiş ortam testçinin yaşamına zarar vermeksizin çoklu kritik koşulların yeniden oluşturulmasına izin vermektedir. Diğer avantajı ise testin yapılabirliğidir. Gerçekte birçok senaryoda belirli test senaryolarının oluşturulması mümkün olmamakla birlikte pahalı ve bazen tehlikelidir. Fakat simüle bir ortamda bunların gerçekleştirilmesi daha mümkün olmaktadır. DDT ile ilgili açıklamadan sonra bu testin geliştirme yaşam döngüsünde ne kadar önemli ve fonksiyonel olduğu görülmektedir.

2.5.5 Hata Enjeksiyon Testi

Yazılım testinde, hata enjeksiyonu testi, bilgisayar sistemlerinin bağımlılığını değerlendirme ve test kapsamını geliştirme için güçlü bir araç olarak kabul edilen bir tekniktir. 1970'lere ait olan ve donanım seviyesinde yapılan kusur enjeksiyonundan gelen bir tekniktir. Donanım uygulamalı kusur enjeksiyonu olarak tanımlanmaktadır.

Bağımlılığa ek olarak; bu teknik, yayılmanın ve sistemdeki kusurların oluşlarının artırılması, performans değerlendirmesi ve sistemin ne kadar kararlı olduğunun doğrulanması için kullanılmaktadır [20].

Genellikle, hata enjeksiyon testi, bileşen arayüzlerinin zayıflığı ve arayüz ve protokoller arası haberleşmeyi kontrol etmek için kullanılan sağlamlık testinin parçası olarak düşünülmektedir.

2.6 Otomatize Yazılım Testi

Bugünlerde, bilgisayar sistemlerde hayatımızda çok çok fazla yer almaktadır ve genelde hayati tehlike arz eden uçak, endüstriyel süreç kontrolü, sağlık, taşıt vb. alanlarda kullanılmaktadır. Günden güne, bu alanlarda kullanıcılar risklere maruz kalmaktadır. Bu yüzden ilgili yazılımın güvenli olduğu garanti edilmelidir. Diğer bir deyişle, güvenliği etkileyebilecek bozuklukları taşımamalıdır.

Son zamanlarda otomatize yazılım testi ile ilgili ciddi çalışmalar gerçekleştirilmektedir ve bu çalışmalar ile test senaryolarının yürütülme zamanlarından tasarruf edilmesi hedeflenmektedir. Ancak, testlerin otomatik hale getirilmesi ile TEY kapsamını ve bağımlılığının arttırıldığı da bir gerçektir. Bu kazançlar çekici görünmesine rağmen, M.N Alam'ın bahsettiği gibi, otomatize yazılım testi genellikle zaman tasarrufu sağlar. Ancak bununla beraber daha çok bağımlı hale gelebilir [22]. Bu durum için birçok şirketin ve araştırmacıların güvenilir teknikler geliştirmekle ilgili ciddi efor harcamasına rağmen, bugünlerde geliştirilen teknikler ancak özel durumlarda kullanılmaktadır. Bu yüzden testlerin otomatik hale getirilebilmesi aşaması, farklı durumlara uyarlanması daha karmaşık olmaktadır. Geliştirme sürecinde esnasında otomatize yazılım testini etkileyebilecek birçok problemten bahsetmek gerekirse [23]:

- Test otomasyonu boş zamanı,
- Net amaçların eksikliği,
- Tecrübe eksikliği,
- Yüksek ciro,
- Çaresizliğe tepki,
- Teknoloji odaklılık,

Bu tip test programlarının oluşturulmasını etkileyen en önemli problemlerden biri de bu programların da yazılım olması ve hatalar tarafından etkilenmesidir. Her şeye rağmen, başarılı bir şekilde uygulamaya geçirildiklerinde hızlı ve yeniden üretilebilir test yürütümü ortaya çıkarmaktadır. Bahsedilen problemler aynı zamanda testçilerin tecrübesi ve olgunlaşmamış test tekniklerinden kaynaklanmaktadır. Test otomasyonuna döndüğümüzde, test otomasyonu diğer programlar üzerinde testleri yürütmek (diğer türlü manuel olarak) için program oluşturan bir geliştirme süreci olarak tanımlanmaktadır.

Test otomasyonu için temelde 2 yaklaşım tanımlanmaktadır:

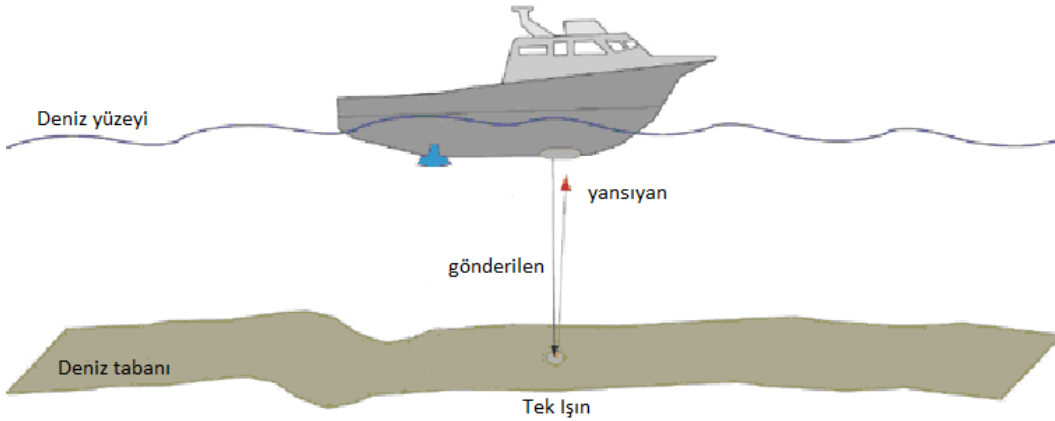
- **Kod güdümlü test:** Beyaz kutu tekniğine dayalı bir test çeşididir. Adından da anlaşılacağı gibi bu test tipinde öncelikle kaynak kodu yazılır ve sonrasında test senaryoları oluşturulur. Kod güdümlü test kaynak kodu doğrulamasına odaklanmaktadır. Bu yüzden fonksiyon, sınıflar, modüller, kütüphaneler vb. gibi yazılım bileşenlerinin testlerine imkan tanımaktadır;
- **Grafiksel kullanıcı arayüz (GKA) testi:** Önceki yaklaşımdan farklı olarak kara kutu tekniğine dayalıdır. Genellikle, GKA ve bazı gelişmiş özellikleri sağlayan otomatize test araçlarına dayanmaktadır. En önemli özelliklerinden biri Kaydet ve Tekrar Çal (Oynat) özelliğidir. Bu özellikte kullanıcıların test senaryolarını oluşturmak için kullanıcı işlemlerini kaydetmesine ve TEY davranışını sağlama amacıyla onları otomatik olarak oynatmasına imkan tanınmaktadır.

3. TEST EDİLEN YAPI

ASELSAN son yıllarda elde ettiği tecrübeler ile değişik sonar sistemlerinin geliştirilmesi çalışmalarına devam etmektedir. Bu çalışma kapsamında ASELSAN tarafından geliştirilen tek ışınlı derinlik ölçüm sonarı KULAÇ Sistemi'nin sistem testleri otomatik hale getirilerek verimlilik sağlanması hedeflenmiştir.

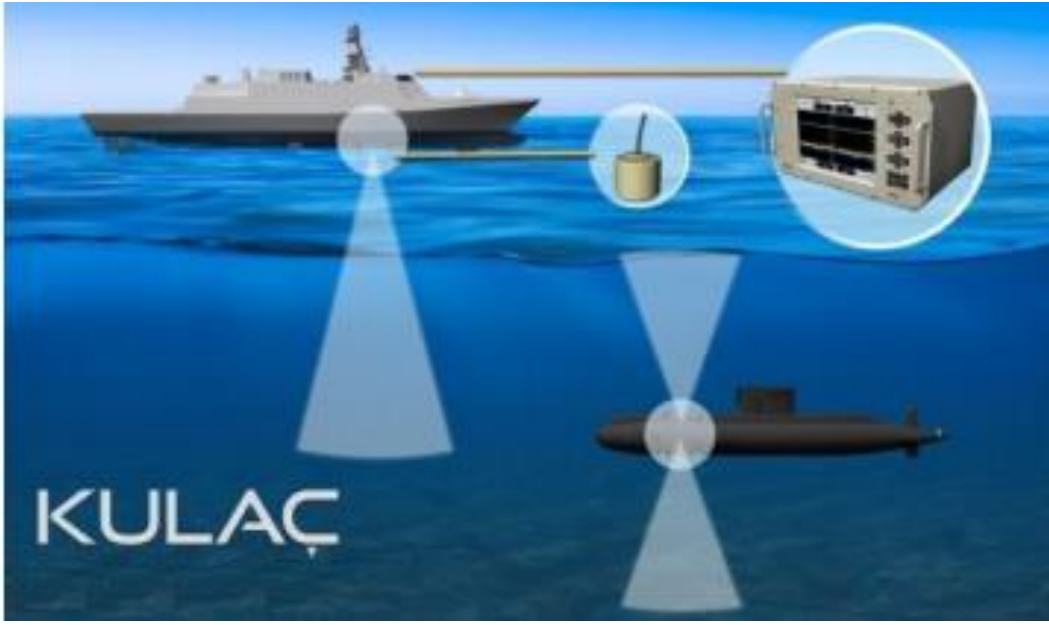
3.1 Sisteme Genel Bakış

Bir seyir yardımcı cihazı olarak askeri deniz platformlarında derinlik ölçümü yapan "Tek Işınlı İskandil" KULAÇ Sistemi, iki veya daha fazla akustik çevirici (transdüser) içeren bir "aktif sonar"dır. Şekil 7'de görüldüğü gibi KULAÇ Sistemi, gemiden yayınlanan akustik sinyalin, dipten veya yüzeyden yansımalarının dinlenmesi ve gönderme/alma zamanı arasındaki farktan hareketle derinlik ölçümü gerçekleştirilmektedir.



Şekil 7. KULAÇ Sistemi Genel Çalışma Senaryosu Gösterimi

Sistemin yapısında; "Gönderme Alma ve İşlemci Birimi", su üstü platformları için 50 kHz ve 200 kHz olmak üzere iki adet; denizaltı platformları için ise iki adet 50 kHz ve iki adet 400 kHz olmak üzere toplam dört adet transdüser ile iki adet "Uzak Gösterge Birimi" bulunmaktadır. Sistem'e ait Gönderme Alma ve İşlemci Birimi ve kullanıcı arayüzü örneği Şekil 8 ve Şekil 9'da gösterilmiştir [33] [34].



Şekil 8. KULAÇ Sistemi Gösterimi



Şekil 9. KULAÇ Sistemi Ekranı (Kullanıcı Arayüz) Gösterimi

4. TEST ALTYAPISI GELİŞTİRME ÇALIŞMALARI

Test altyapısı geliştirme çalışmaları; KULAÇ Derinlik Ölçüm Sonarı Sistemi için; sistem doğrulama, üretim ve regresyon testlerinin gerçekleştirilmesi amacıyla sisteme ait manuel testlerin oda koşullarında/laboratuvar ortamında yürütülmesi ile ilgili yapılan hazırlık faaliyetlerini içermektedir. Sonraki aşamada ise gerçekleştirilen manuel testlerin otomatize edilmesi için tasarlanan ve geliştirilen otomatik test altyapısına yönelik çalışmalar belirtilecektir.

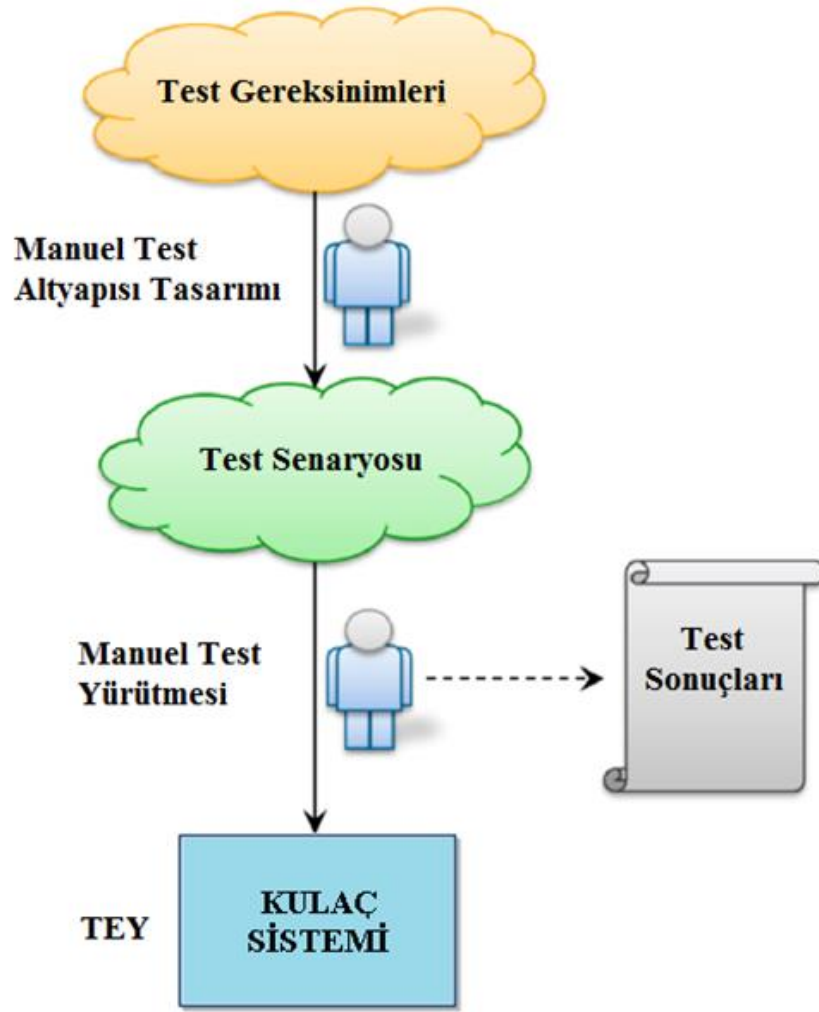
Bu çalışmada kullanılan KULAÇ Sistemi savunma sanayinde kullanılan bir sistem olduğu için, sistemin mesajlaşma arayüzleri ve kullanıcı arayüzleri hakkında detaylı bilgi verilmeyecek olup, çevre alt sistem birimleri için ve uygulanan test yaklaşımına yönelik genel bilgiler verilecektir.

Tez çalışmasında KULAÇ Sistemi için Test Edilen Yapı (TEY) ifadesi kullanılacaktır.

4.1 Manuel Test Altyapısının Oluşturulması

Manuel test, sistemlerin testleri sırasında, test operatörünün son kullanıcı gibi davranarak, sistemdeki hataları ve meydana gelebilecek istenmeyen durumları, herhangi bir test otomasyonu ve test betiği kullanmadan test etmesidir [24]. Manuel test, hataları bulmak için yürütülen en genel süreçtir.

Sistemlerin manuel test ile doğrulanması esnasında, her bir gereksinim maddesini doğrulayacak test tanımı senaryoları oluşturularak, testlerin yürütülmesi sağlanmaktadır. Her bir test adımının tamamlanmasının ardından elde edilen test sonuçları, test operatörleri tarafından test sonuçları şeklinde raporlanır. Bu sonuçlar beklenen sonuçlar ile karşılaştırılarak test başarımları değerlendirilir. Böylelikle manuel test süreci tamamlanmış olur. Şekil 10'da manuel test senaryosuna ait akış şeması görülmektedir.



Şekil 10. Manuel Test Senaryosu

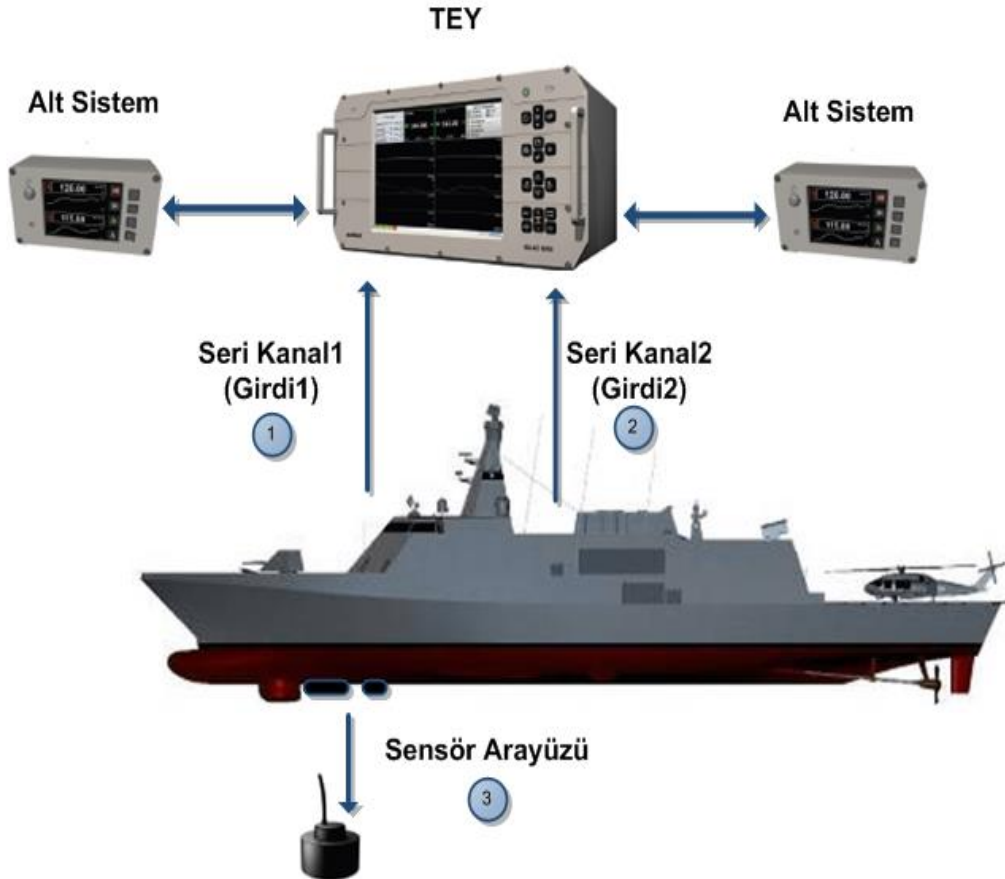
Günümüzde sistemlerin müşteriye daha güvenilir ve hızlı ulaşmasını sağlamak amacıyla, manuel testlerin gerçekleştirilmesi sırasında testlerin daha etkin şekilde kurgulanması, çevre birimlerin simüle edilmesi fikri önem kazanmıştır [9]. Bu bakımdan projelerde, çevre birimlerin (alt sistemlerin) hazır olmadığı durumlarda da testlerin gerçekleştirilebilmesi amacıyla çevre birimlerin simüle edilmesi, test faaliyetleri açısından önemli bir adım olarak göze çarpmaktadır.

TEY'e yönelik manuel test altyapısının kurularak testlerin gerçekleştirilebilmesi hususunda yapılan çalışmalar Bölüm 4.1.1'de anlatılmaktadır.

4.1.1 Çevre Birimlerin Simüle Edilmesi

Test faaliyetleri gerçekleştirilmesi planlanan TEY için, sistem testlerinin gemi platformlarında gerçek donanımlarla test edilmesi hem maliyetli bir çözüm olarak görülmüş; hem de gerçekleştirilen aktiviteler sonucunda yapılan çalışmaların verimsiz olduğu gözlenmiştir. Bu nedenle TEY için gerçekleştirilecek sistem doğrulama testlerinin, çevre ortamlara bağlı olmaksızın oda koşullarında/ laboratuvar ortamında test edilebilmesi gerekliliği sonucuna varılmış olup, bu kapsamda TEY'e yönelik alt sistem/çevre birimlerin simüle edilmesine yönelik çalışmalar gerçekleştirilmiştir.

Sistem testlerinin manuel olarak laboratuvar ortamında gerçekleştirilebilmesi amacıyla, TEY'e yönelik simüle edilecek çevre birimler Şekil 11'de gösterilmiş olup, ileriki bölümde 1,2 ve 3 olarak numaralandırılmış çevre birimlerin hangi arayüzler olduğu ve test altyapısının kurulması için yapılan çalışmalar açıklanmıştır.



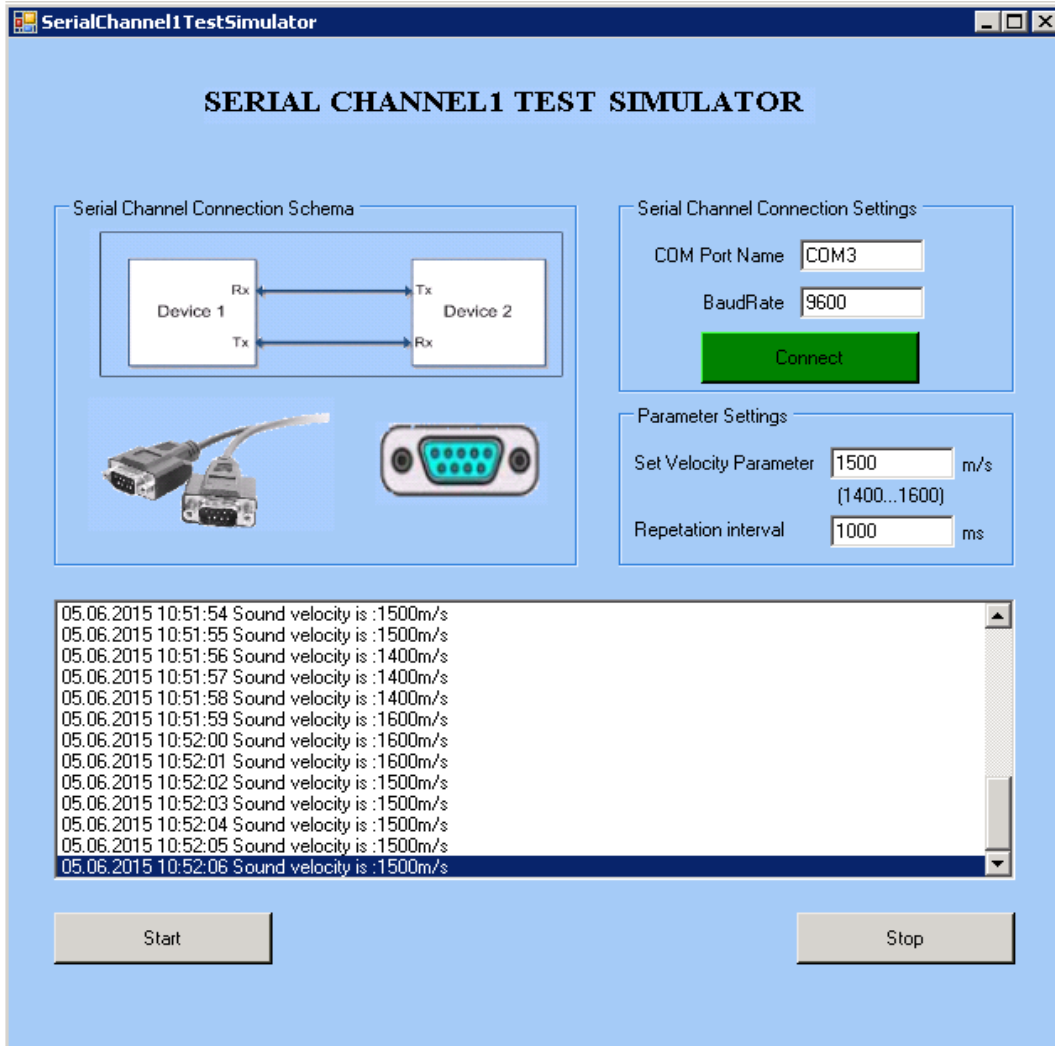
Şekil 11. TEY İçin Çevre Birim/Alt Sistemlerin Gösterimi

4.1.1.1 Seri Kanal1 Alt Sistem Arayüzünün Simüle Edilmesi

TEY'in sahip olduğu Seri Kanal1 alt sistemi, derinlik ölçümü sırasında kullanılan ses hızı parametresinin hesaplanmasını sağlayan ve ilgili verinin kullanılması amacıyla sisteme girdide bulunan bir alt sistemdir.

Akustik dalgaların su ortamında normal koşullardaki yayılma hızı 1500 m/s olarak kabul edilmekte olup; ses hızı; suyun basıncı, tuzluluğu, sıcaklığı ve derinliği ile değişen bir parametredir [25].

TEY'e yönelik sistem testlerinin laboratuvar ortamında gerçekleştirilebilmesi amacıyla, TEY'in sahip olduğu Seri Kanal1 alt sistemi arayüzü, haberleşme veri formatına uygun olarak C# yazılım geliştirme ortamında yazılımsal olarak simüle edilmiş olup, ilgili simülatöre yönelik ekran görüntüsü Şekil 12'de belirtilmiştir.



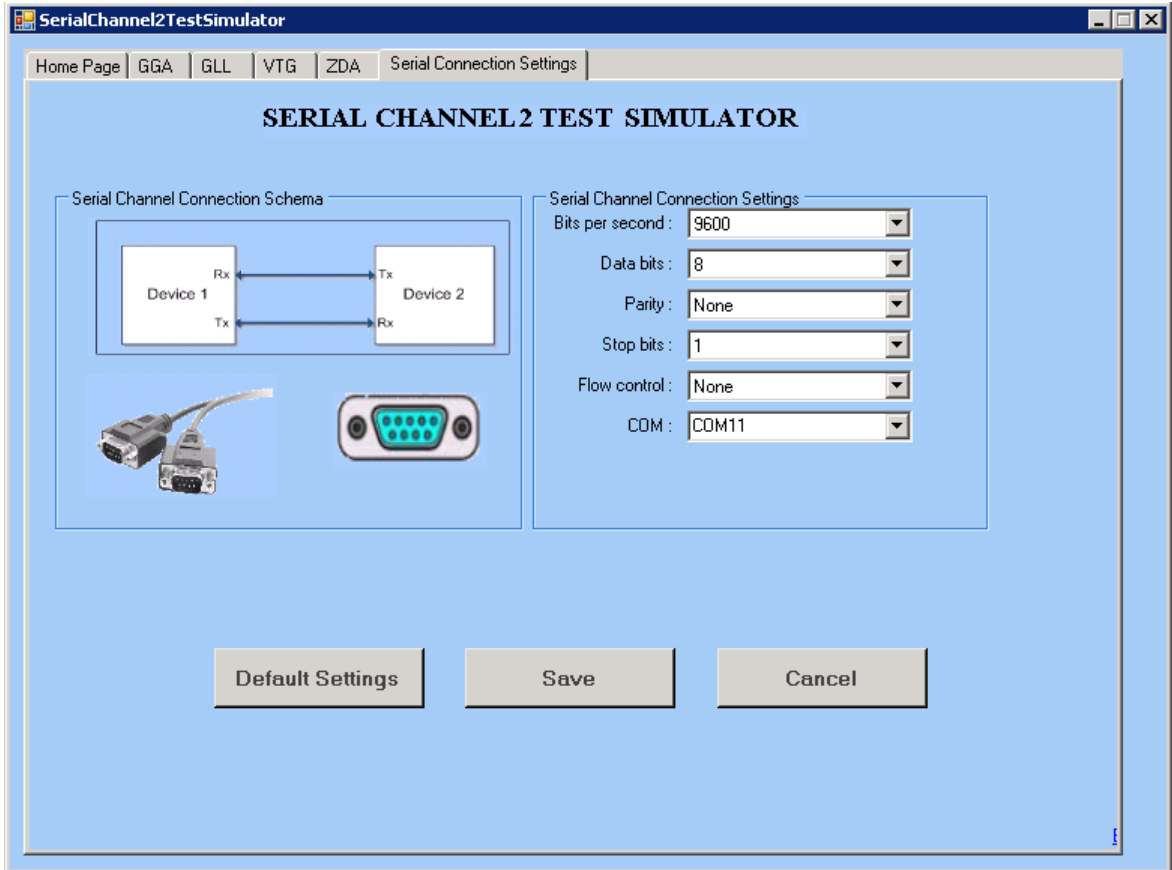
Şekil 12. Seri Kanal1 Alt Sistemi Arayüz Simülatörü

4.1.1.2 Seri Kanal2 Alt Sistem Arayüzünün Simüle Edilmesi

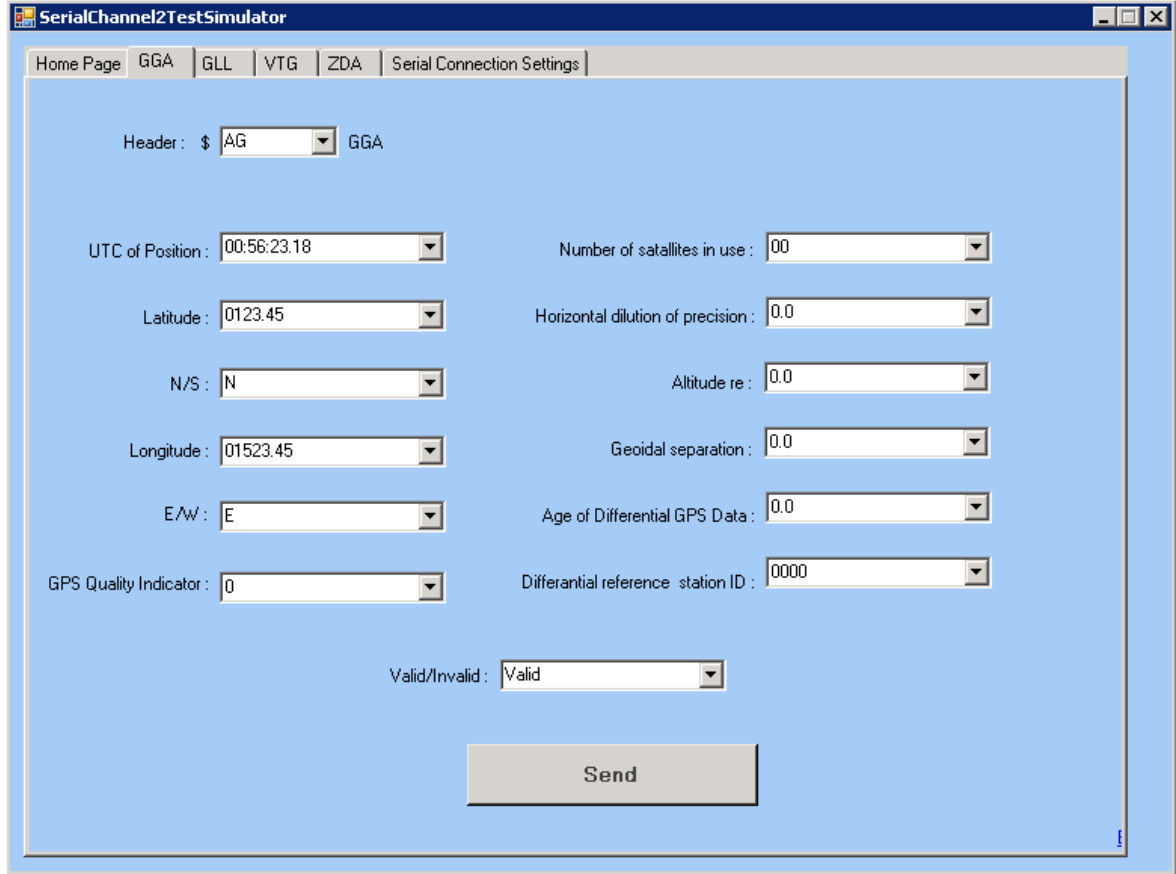
Genel olarak denizcilik ve askeri alanlarda kullanılan ve elektronik cihazların birbiriyle iletişimini sağlayan Seri Kanal2 alt sistemi; seri iletişim protokolüne uygun olarak sisteme enlem, boylam, hız ve zaman gibi verileri uygun formatta gönderen bir alt sistemdir.

TEY'in sahip olduğu Seri Kanal2 alt sistemi arayüzü C# yazılım geliştirme ortamı kullanılarak simüle edilmiş olup, geliştirilen simülatör yazılımının ekran görüntüsü Şekil 13 ve Şekil 14'te yer almaktadır.

Seri Kanal2 Alt Sistemi Simülatör Yazılımı, 4 farklı mesaj tipi için veri göndermek üzere tasarlanmış olup, belirlenmiş periyotlarla gönderilen mesajların içerikleri kullanıcı ara yüzünden değiştirilebilmektedir. Ayrıca bu mesajlar periyodik olarak da gönderilmektedir.



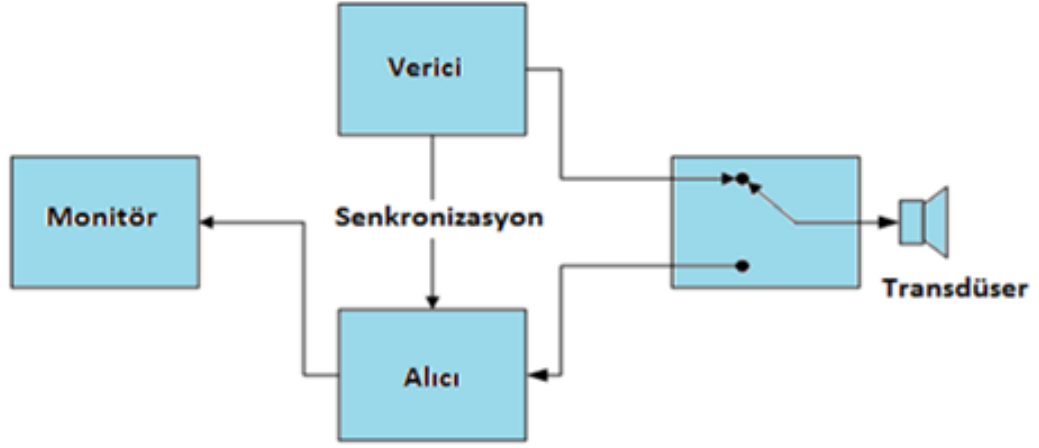
Şekil 13. Seri Kanal2 Alt Sistemi Simülatör Yazılımı Ekran Görüntüsü



Şekil 14. Seri Kanal2 Alt Sistemi Simülör Yazılımı Alt Ekran Sekmesi Görüntüsü

4.1.1.3 Sensör Alt Sistem Arayüzünün Simüle Edilmesi

TEY'in sahip olduğu sensör alt sistemi, sesin suda yayılması prensibiyle çalışmaktadır. İlgili sensör alt sistemi, öncelikle elektrik enerjisini ses enerjisine çevirdikten sonra, ses dalgasının suda bulunan herhangi bir cisim ya da deniz dibinden yansması sonucu, yansıyan ses enerjisini, elektrik enerjisine tekrar geri dönüştüren sistemlerdir [32]. (Bkz. Şekil 15)



Şekil 15. Sensör Alt Sistemi Blok Şeması

TEY, Bölüm 3'de de belirtildiği üzere, düşük frekans ve yüksek frekans olmak üzere 2 adet sensör alt sistem arayüzünden oluşan bir sistemdir.

TEY'e yönelik olarak oda koşullarında manuel test ortamının kurulabilmesi ve testlerin başarılı olarak gerçekleştirilmesi için; sistemin sensör alt sistemi arayüzlerinin doğru bir şekilde simüle edilmesi önem arz etmektedir. Fakat sisteme ait sensör alt sistemleri akustik sensörler olduğu ve ancak su altında çalışabildiği için bu aşamada sistem testlerinin laboratuvar ortamında gerçekleştirilebilmesi hususunda problemler meydana gelmiştir.

Bu kapsamda, sensör alt sistemi arayüzünün yalnızca yazılımsal olarak simüle edilmesi yaklaşımı değerlendirilmiştir. Ancak ilgili yaklaşımın, sistemin alıcı verici donanım altyapısına yönelik elektriksel doğrulamaları içermeyeceği için, bu görüşün sistem testleri esnasında yetersiz kalacağı değerlendirilmiştir.

Tüm bu bilgiler ışığında, sistem test faaliyetlerinin oda koşullarında doğru ve eksiksiz olarak test edilebilmesi için, sistemin dip derinliğini ölçebilmesi amacıyla gerçek ortamı (deniz ortamını) simüle eden, elektriksel zaman gecikmesi üreterek, alıcı verici donanım altyapısını da doğrulayan donanımsal bir öykünücü (emülatör) ihtiyacının olduğu sonucuna ulaşılmıştır.

Belirtilen özellikteki donanımının (öykünücünün), firmanın kendi imkânlarıyla geliştirilmesinin, proje açısından zaman olarak gecikmeye sebep olacağı değerlendirilerek, konu ile ilgili hazır ticari ürünlerin var olup olmadığına yönelik

piyasada arařtırmalar yapılmıřtır. Yapılan arařtırmalar sonucunda, Electronic Devices Inc. firmasının iskandil sistemlerinde bakım-onarım amaçlı olarak kullanılması amacıyla tasarlamıř olduėu “Derinlik Ölçüm Test Seti” ürünlerinin, piyasada var olduėu görülmüřtür [28]. Manuel testlerin oda kořullarında gerçekteřtirilebilmesi amacıyla, düşük frekans sensör arayüzü için EDI firmasının DSTS-4A, yüksek frekans sensör arayüzü için DSTS-5A modeli olan “Derinlik Ölçüm Test Seti” ürünlerinin öykünücü olarak kullanılabilereceėi sonucu ortaya çıkmıřtır. (Bkz. Őekil 16-Őekil 17)



Őekil 16. EDI DSTS-4A Derinlik Ölçüm Test Seti (Düşük Frekans)



Őekil 17. EDI DSTS-5A Derinlik Ölçüm Test Seti (Yüksek Frekans)

Manuel testlerin oda kořullarında gerçekteřtirilebilmesi amacıyla kullanılan Derinlik Ölçüm Test Seti ürünlerinin temel özellikleri ařaėıda belirtilmiřtir:

- Sensör alt sistemi ile benzer özelliklerde yük karakteristiğine sahip olma,
- Gelen sinyali tespit etme ve sinyalini frekans, darbe genişliği, darbe periyodu ve darbe genliğini belirlenmesi şeklinde kullanıcı arayüzüne aktarma,
- Gelen sinyale göre programlanan derinlikte tespit için karşı cevap sinyali üretme,
- Gönderilecek sinyalin frekansını, yansıma derinliğini, sinyal genişliğini kullanıcı arayüzünden değiştirme/ayarlama

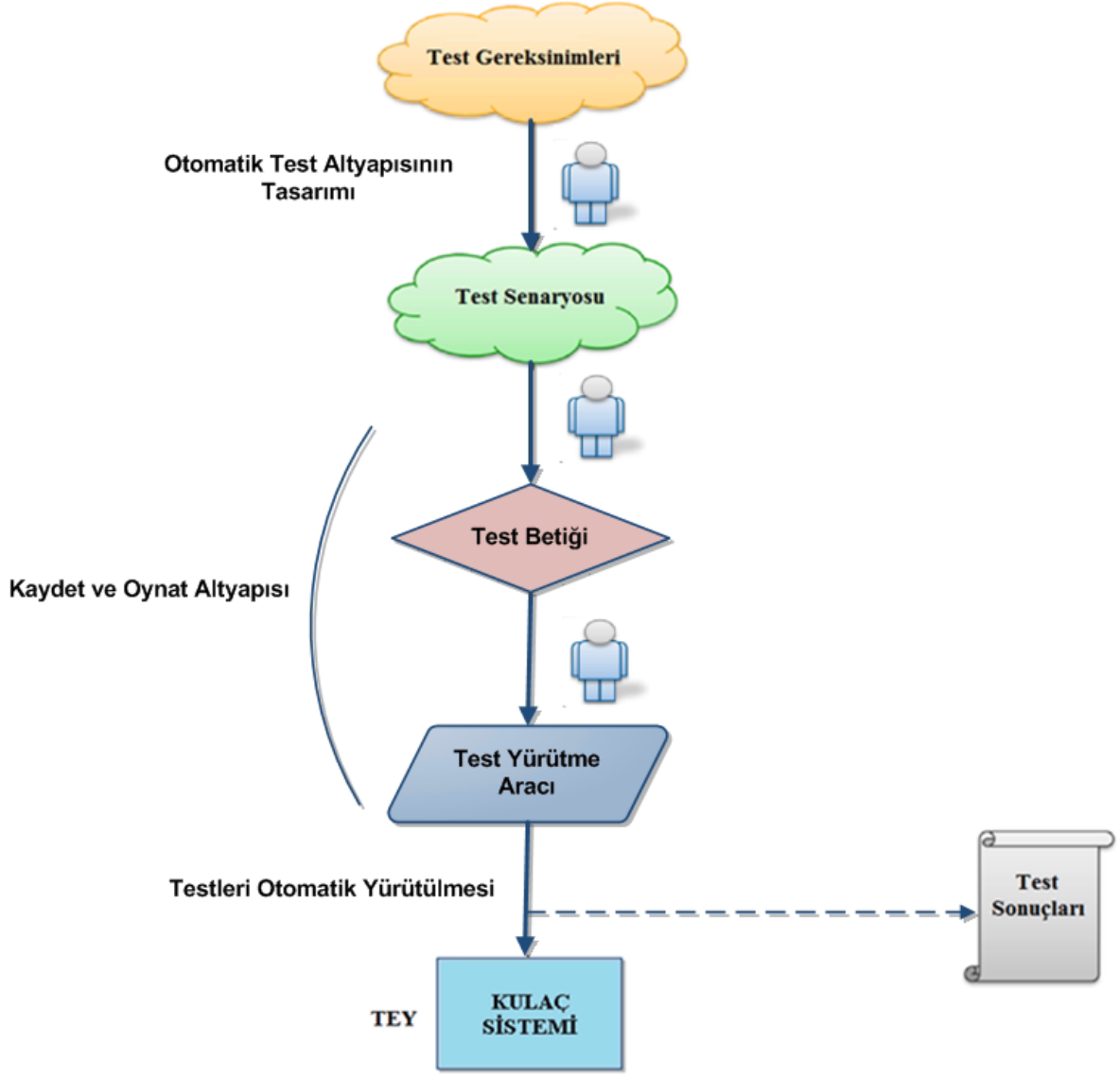
Yukarıda belirtilen “Derinlik Ölçüm Test Seti” ürünlerinden ileriki bölümlerde derinlik simülatörü olarak bahsedilecektir.

Yapılan çevre birimlerin simüle edilmesi çalışması ile sisteme özel Karaya Konuşlu Test Altyapısı oluşturulmuştur. Oluşturulan test sistemi ile üretilen sistemin senaryosal olarak doğrulanması, sistem testlerinin laboratuvar ortamına taşınması sağlanmış olup, bu kapsamda iş gücü, zaman ve test maliyeti gibi etkenler konusunda verimlilik sağlanmıştır.

4.2 Otomatik Test Altyapısının Oluşturulması

Otomatik test, sistemlerin testleri sırasında, test operatörü olmadan, test araçları ve test betiği kullanarak, kaydedilen senaryolara uygun olarak testlerin gerçekleştirilmesi işlemidir.

Sistemlerin otomatik test ile doğrulanması sırasında, her bir gereksinim maddesini doğrulayacak test tanımı senaryoları test betiği kullanılarak oluşturulur ve test otomasyon araçları kullanılarak test senaryoları kayıt edilir. Daha sonra kaydedilen senaryolara uygun olarak test yürütme araçları yardımıyla, testlerin tamamlanması sağlanır. Test faaliyetleri sonucunda test başarımları, beklenen sonuçlarla gerçek sonuçlar karşılaştırılarak, otomatik olarak oluşmaktadır [29]. Otomatik test senaryosuna ilişkin blok şema Şekil 18’de yer almaktadır.

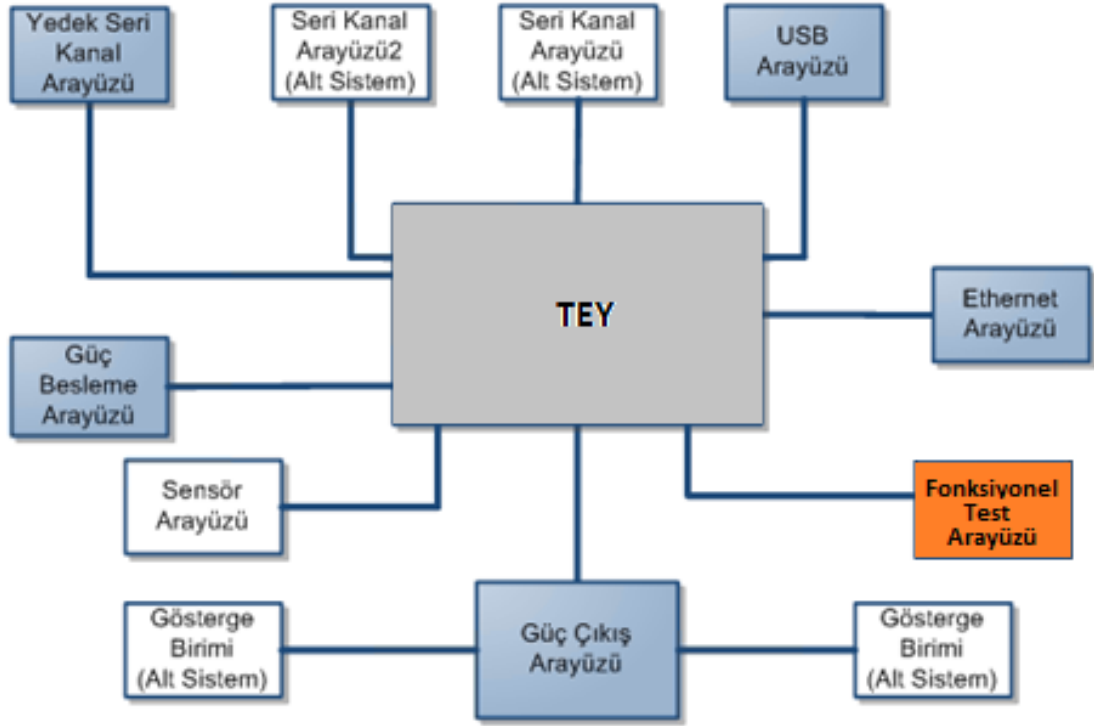


Şekil 18. Otomatik Test Senaryosu

TEY'e ait tüm arayüzler Şekil 19'da görülmektedir. Yukarıda manuel test altyapısı oluşturma faaliyetinde açıklanan ve simüle edilen arayüzler beyaz renkli kutular olarak gösterilmektedir.

Mavi ve turuncu renkli kutularda gösterilen arayüzler ise, otomatik test altyapısı geliştirme faaliyeti kapsamında otomatize edilen arayüzleri simgelemektedir. Ethernet arayüzü, USB arayüzü, yedek seri kanal arayüzü, güç besleme arayüzü ve güç çıkış arayüzü; fiziksel arayüzler olarak adlandırılmış olup, mavi renk ile gösterilmiştir. Turuncu renkli kutu ile belirtilen fonksiyonel test arayüzü ise sistemin çalışma senaryosuna uygun olarak grafiksel kullanıcı arayüzünün test edilmesi

olarak belirtilmektedir. Bu arayüzlerin testlerine ilişkin detaylı açıklamalar sonraki bölümde anlatılmıştır.



Şekil 19. TEY Arayüz Blok Şeması

Bu kapsamda TEY'e yönelik sistem testlerinin otomatizasyonu için, fiziksel arayüzlere yönelik testlerin ve senaryolara uygun olarak grafiksel kullanıcı arayüzünün fonksiyonel olarak test edilmesi gerekmektedir [29].

4.2.1 Fiziksel Arayüz Testlerinin Otomatize Edilmesi

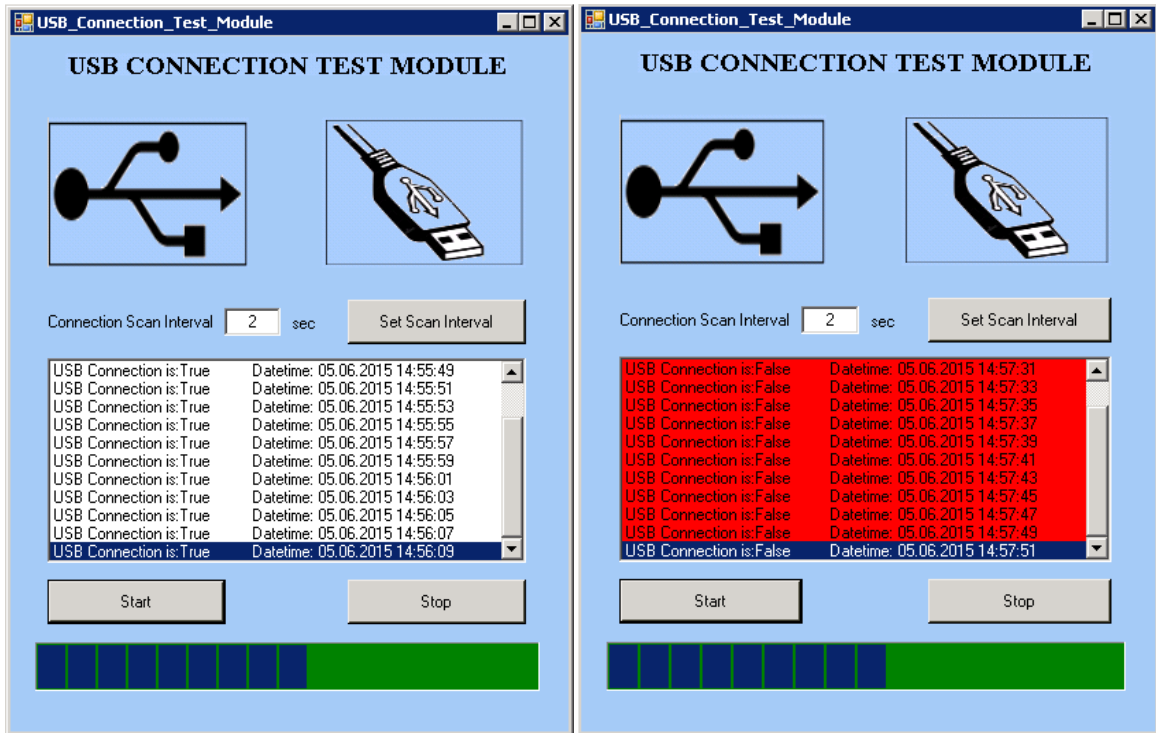
Sisteme ait fiziksel arayüzler ve test ediliş amaçları aşağıda belirtilmiş olup; testlerin otomatik olarak gerçekleştirilmesi için fiziksel arayüzlere yönelik olarak C# yazılım geliştirme ortamı kullanılarak modüler bir donanım/yazılım test altyapısı tasarımı gerçekleştirilmiştir.

4.2.1.1 USB Bağlantı Arayüzü

USB, dilimizde “Evrensel Seri Veri Yolu” olarak adlandırılmakta olup, farklı çevre birimlerin bilgisayar bağlantılı kullanımlarını sağlayan seri yapıllı bir bağlantı biçimidir. Evrensel Seri Veri Yolu, tak-çalıştır özelliği sayesinde bir çok cihaz tarafından kolaylıkla kullanılmasının yanı sıra, iki bağlantı arasında 5 metre mesafeye kadar iletişim imkanı sunmaktadır [31].

TEY'in sahip olduğu Evrensel Seri Veri Yolu fiziksel arayüzü, sistemin kayıt parametrelerinin dışa aktarımını sağladığı gibi, sistem üzerinde uygulanacak güncelleme faaliyetlerinin de başarıyla gerçekleştirilmesi açısından önem arz etmektedir.

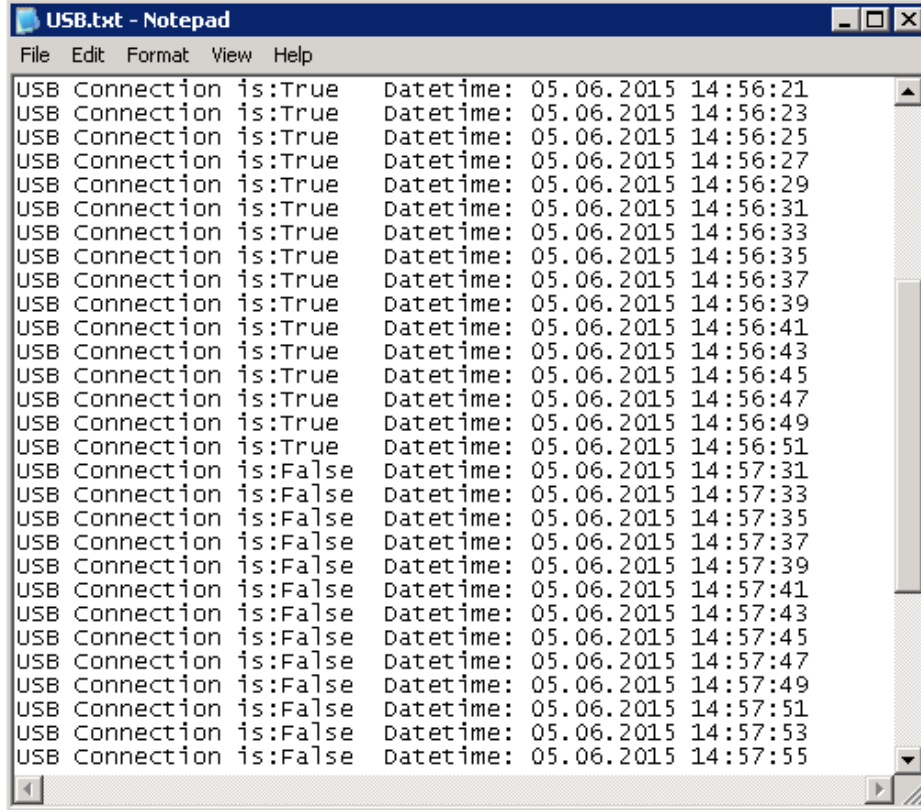
Bu kapsamda; sistem testlerinin otomatize edilmesi amacıyla, TEY'in sahip olduğu USB fiziksel arayüzünün çalışıp çalışmadığının kontrolü ve test sonuçlarının kayıt altına alınması C# yazılım geliştirme ortamında USB bağlantı test modülü simülatörü yazılarak sağlanmıştır.



Şekil 20. USB Bağlantı Test Modülü Yazılımı Ekran Görüntüsü

C# ortamında geliştirilen USB bağlantı test modülü ekran görüntüleri Şekil 20'de sunulmaktadır. Şeklin sol tarafında otomatik olarak test edilen USB arayüzüne ait

testin başarıyla tamamlandığı ekran görüntüsü yer alırken, sağ taraftaki görüntü otomatik yapılan testin başarısız sonuçlandığını göstermektedir. Diğer yandan, Şekil 21’de USB arayüzüne ait yapılan otomatik testin test sonuçlarının kayıt altına alındığını gösteren text (.txt) dosyasına ait ekran görüntüsü yer almaktadır.



```
USB.txt - Notepad
File Edit Format View Help
USB Connection is:True Datetime: 05.06.2015 14:56:21
USB Connection is:True Datetime: 05.06.2015 14:56:23
USB Connection is:True Datetime: 05.06.2015 14:56:25
USB Connection is:True Datetime: 05.06.2015 14:56:27
USB Connection is:True Datetime: 05.06.2015 14:56:29
USB Connection is:True Datetime: 05.06.2015 14:56:31
USB Connection is:True Datetime: 05.06.2015 14:56:33
USB Connection is:True Datetime: 05.06.2015 14:56:35
USB Connection is:True Datetime: 05.06.2015 14:56:37
USB Connection is:True Datetime: 05.06.2015 14:56:39
USB Connection is:True Datetime: 05.06.2015 14:56:41
USB Connection is:True Datetime: 05.06.2015 14:56:43
USB Connection is:True Datetime: 05.06.2015 14:56:45
USB Connection is:True Datetime: 05.06.2015 14:56:47
USB Connection is:True Datetime: 05.06.2015 14:56:49
USB Connection is:True Datetime: 05.06.2015 14:56:51
USB Connection is:False Datetime: 05.06.2015 14:57:31
USB Connection is:False Datetime: 05.06.2015 14:57:33
USB Connection is:False Datetime: 05.06.2015 14:57:35
USB Connection is:False Datetime: 05.06.2015 14:57:37
USB Connection is:False Datetime: 05.06.2015 14:57:39
USB Connection is:False Datetime: 05.06.2015 14:57:41
USB Connection is:False Datetime: 05.06.2015 14:57:43
USB Connection is:False Datetime: 05.06.2015 14:57:45
USB Connection is:False Datetime: 05.06.2015 14:57:47
USB Connection is:False Datetime: 05.06.2015 14:57:49
USB Connection is:False Datetime: 05.06.2015 14:57:51
USB Connection is:False Datetime: 05.06.2015 14:57:53
USB Connection is:False Datetime: 05.06.2015 14:57:55
```

Şekil 21. USB Bağlantı Test Sonuçları Ekran Görüntüsü

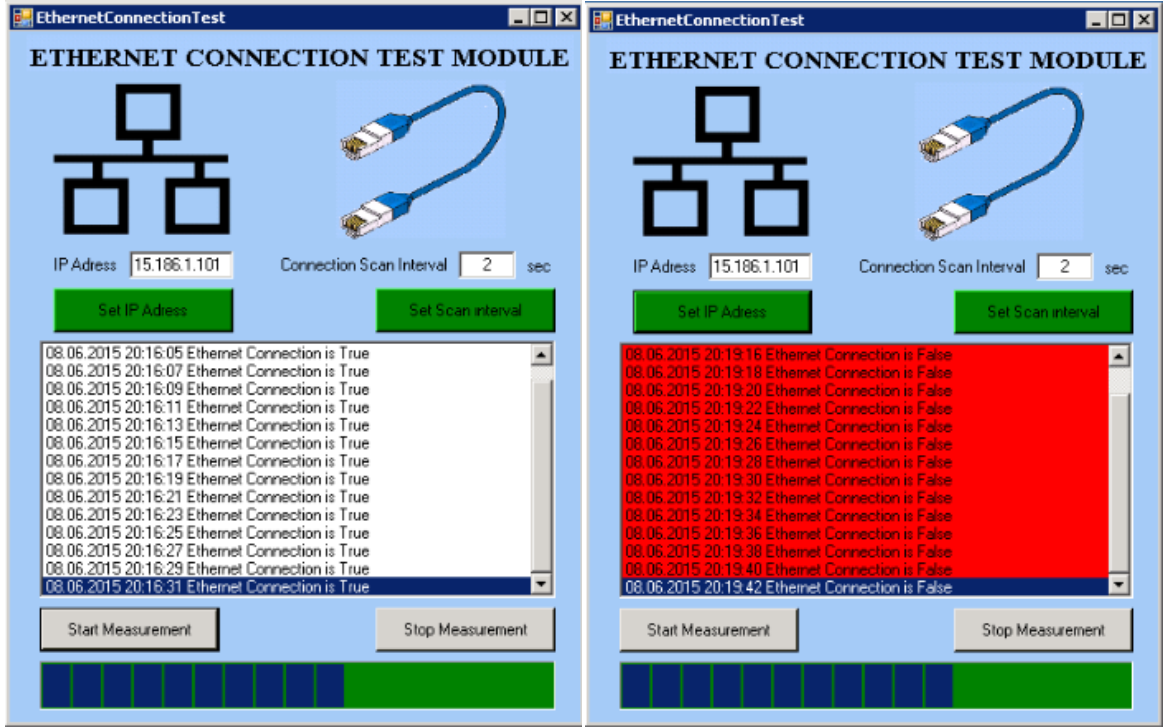
4.2.1.2 Ethernet Bağlantı Arayüzü

Ethernet, yerel alan ağları (LAN) ve metropolitan alan ağları (MAN) için bilgisayar ağ teknoloji ailesi olarak tanımlanmakta olup; farklı çevre birimlerin aynı çalışma ortamında birbiriyle veri alışverişinde bulunması için bağlantı kurmalarını sağlayan bir bağlantı biçimidir [31].

TEY'in sahip olduğu Ethernet fiziksel arayüzü, sistemin aynı çalışma ortamında bulunan CPU, disk gibi kaynaklarının ve yazıcı vb. gibi farklı çevre birimlerinin birbiriyle haberleşmesi sağlayan bir standart haberleşme arayüzüdür.

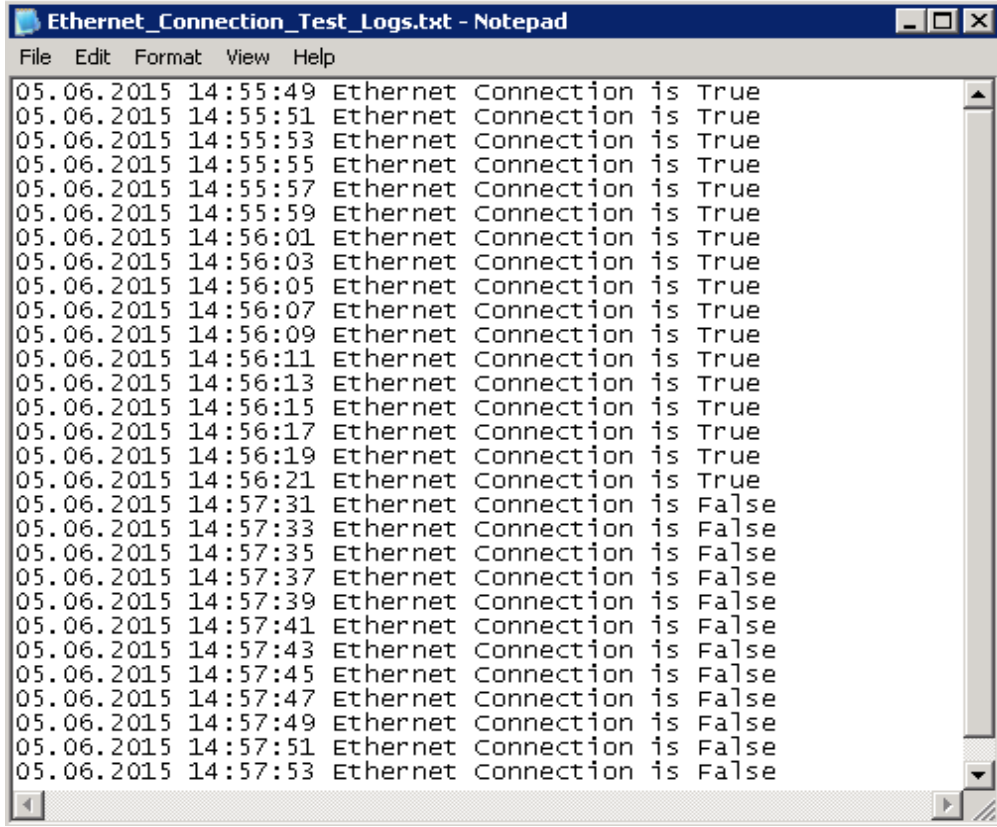
Bu kapsamda; sistem testlerinin otomatize edilmesi amacıyla, TEY'in sahip olduğu ethernet fiziksel arayüzünün çalışıp çalışmadığının kontrolü ve test sonuçlarının

kayıt altına alınması C# yazılım geliştirme ortamında ethernet bağlantı test modülü simülatörü yazılarak sağlanmıştır.



Şekil 22. Ethernet Bağlantı Test Modülü Yazılımı Ekran Görüntüsü

C# ortamında geliştirilen ethernet bağlantı test modülü ekran görüntüleri Şekil 22'de sunulmaktadır. Şeklin sol tarafında otomatik olarak test edilen ethernet bağlantı arayüzüne ait testin başarıyla tamamlandığı ekran görüntüsü yer alırken, sağ taraftaki görüntü otomatik yapılan testin başarısız sonuçlandığını göstermektedir. Diğer yandan, Şekil 23'te ethernet bağlantı arayüzüne ait yapılan otomatik testin test sonuçlarının kayıt altına alındığını gösteren text (.txt) dosyasına ait ekran görüntüsü yer almaktadır.



```
Ethernet_Connection_Test_Logs.txt - Notepad
File Edit Format View Help
05.06.2015 14:55:49 Ethernet Connection is True
05.06.2015 14:55:51 Ethernet Connection is True
05.06.2015 14:55:53 Ethernet Connection is True
05.06.2015 14:55:55 Ethernet Connection is True
05.06.2015 14:55:57 Ethernet Connection is True
05.06.2015 14:55:59 Ethernet Connection is True
05.06.2015 14:56:01 Ethernet Connection is True
05.06.2015 14:56:03 Ethernet Connection is True
05.06.2015 14:56:05 Ethernet Connection is True
05.06.2015 14:56:07 Ethernet Connection is True
05.06.2015 14:56:09 Ethernet Connection is True
05.06.2015 14:56:11 Ethernet Connection is True
05.06.2015 14:56:13 Ethernet Connection is True
05.06.2015 14:56:15 Ethernet Connection is True
05.06.2015 14:56:17 Ethernet Connection is True
05.06.2015 14:56:19 Ethernet Connection is True
05.06.2015 14:56:21 Ethernet Connection is True
05.06.2015 14:57:31 Ethernet Connection is False
05.06.2015 14:57:33 Ethernet Connection is False
05.06.2015 14:57:35 Ethernet Connection is False
05.06.2015 14:57:37 Ethernet Connection is False
05.06.2015 14:57:39 Ethernet Connection is False
05.06.2015 14:57:41 Ethernet Connection is False
05.06.2015 14:57:43 Ethernet Connection is False
05.06.2015 14:57:45 Ethernet Connection is False
05.06.2015 14:57:47 Ethernet Connection is False
05.06.2015 14:57:49 Ethernet Connection is False
05.06.2015 14:57:51 Ethernet Connection is False
05.06.2015 14:57:53 Ethernet Connection is False
```

Şekil 23. USB Bağlantı Test Sonuçları Ekran Görüntüsü

4.2.1.3 Güç Giriş Arayüzü

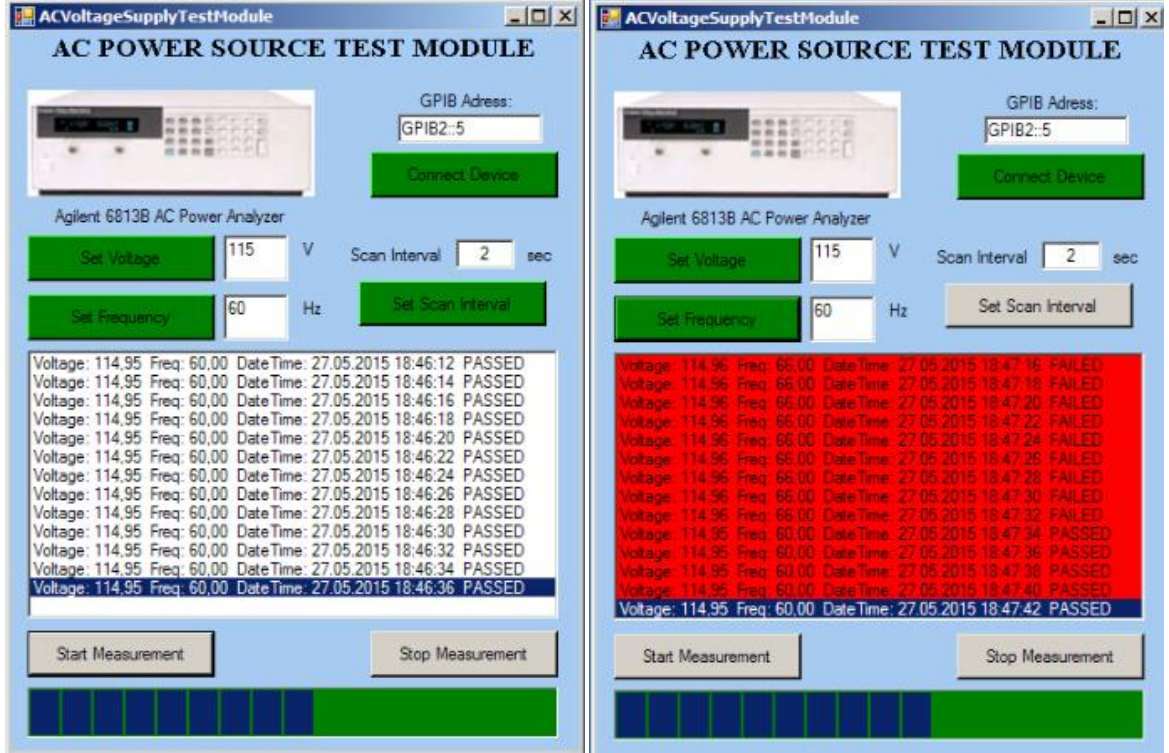
TEY'in çalışması için gerekli besleme gerilimi 115 VAC ve frekansı 60 Hz'dir. Şebeke (priz) geriliminin bu özellikleri sağlayamamasından dolayı, TEY'e yönelik gerçekleştirilen test doğrulama faaliyetleri sırasında AC güç kaynağı cihazının kullanılması gerekmektedir.

AC güç kaynağı, tipik bir şebeke voltajını alarak, bazı filtreleme işlemleri ile birlikte gerilim seviyesini istenen seviyelere indirmektedir.

İlgili testler sırasında Agilent firmasının 6813B Veri AC Güç Kaynağı Analizörü cihazı kullanılmıştır [27].

Sistem testlerinin otomatize edilmesi amacıyla, TEY'in sahip olduğu güç giriş arayüzünün akım, gerilim ve frekans seviyelerinin testler süresinde sürekli olarak kontrol edilmesi ve test sonuçlarının kayıt altına alınması C# yazılım geliştirme ortamında AC güç kaynağı test modülü simülatörü yazılarak sağlanmıştır.

İlgili test simülatörünün güç kaynağı analizörü ile haberleşmesi için genel amaçlı arayüz veri yolu (GPIB), IEEE-488 haberleşme protokolü kullanılmıştır [31].



Şekil 24. AC Güç Kaynağı Test Modülü Yazılımı Ekran Görüntüsü

C# ortamında geliştirilen AC güç kaynağı test modülü ekran görüntüleri Şekil 24'te sunulmaktadır. Şeklin sol tarafında otomatik olarak test edilen AC güç kaynağı test arayüzüne ait testin başarıyla tamamlandığı ekran görüntüsü yer alırken, sağ taraftaki görüntü otomatik yapılan testin başarısız sonuçlandığını göstermektedir. Diğer yandan, Şekil 25'te AC güç kaynağı test arayüzüne ait yapılan otomatik testin test sonuçlarının kayıt altına alındığını gösteren text (.txt) dosyasına ait ekran görüntüsü yer almaktadır.

Voltage	Current	Freq	Pow	DateTime	Result
114,9520	0,4909	60,0000	52,4561	27.05.2015 18:46:12	PASSED
114,9510	0,4913	60,0000	52,3557	27.05.2015 18:46:14	PASSED
114,9530	0,4921	60,0000	52,5588	27.05.2015 18:46:16	PASSED
114,9520	0,4899	60,0000	52,7668	27.05.2015 18:46:18	PASSED
114,9530	0,4871	60,0000	52,6716	27.05.2015 18:46:20	PASSED
114,9510	0,4879	60,0000	52,3282	27.05.2015 18:46:22	PASSED
114,9520	0,4893	60,0000	52,2384	27.05.2015 18:46:24	PASSED
114,9520	0,4909	60,0000	52,5376	27.05.2015 18:46:26	PASSED
114,9520	0,4911	60,0000	52,5516	27.05.2015 18:46:28	PASSED
114,9530	0,4885	60,0000	52,4131	27.05.2015 18:46:30	PASSED
114,9520	0,4881	60,0000	52,3191	27.05.2015 18:46:32	PASSED
114,9520	0,4913	60,0000	52,4416	27.05.2015 18:46:34	PASSED
114,9520	0,4899	60,0000	52,5878	27.05.2015 18:46:36	PASSED
114,9520	0,4873	60,0000	52,3598	27.05.2015 18:46:38	PASSED
114,9530	0,4907	60,0000	52,5286	27.05.2015 18:46:40	PASSED
114,9530	0,4888	60,0000	52,4422	27.05.2015 18:46:42	PASSED
114,9530	0,4881	60,0000	52,3038	27.05.2015 18:46:44	PASSED
114,9530	0,4900	60,0000	52,5628	27.05.2015 18:46:46	PASSED
114,9580	0,4917	66,0000	52,7801	27.05.2015 18:47:16	FAILED
114,9570	0,4949	66,0000	52,2697	27.05.2015 18:47:18	FAILED
114,9580	0,4920	66,0000	52,1743	27.05.2015 18:47:20	FAILED
114,9580	0,4940	66,0000	52,6142	27.05.2015 18:47:22	FAILED
114,9570	0,4945	66,0000	52,0846	27.05.2015 18:47:24	FAILED
114,9570	0,4924	66,0000	52,3027	27.05.2015 18:47:26	FAILED
114,9560	0,4951	66,0000	52,5761	27.05.2015 18:47:28	FAILED
114,9570	0,4924	66,0000	52,1517	27.05.2015 18:47:30	FAILED
114,9570	0,4924	66,0000	52,1858	27.05.2015 18:47:32	FAILED
114,9530	0,4909	60,0000	52,5916	27.05.2015 18:47:34	PASSED
114,9520	0,4876	60,0000	52,2249	27.05.2015 18:47:36	PASSED
114,9530	0,4861	60,0000	52,2970	27.05.2015 18:47:38	PASSED
114,9540	0,4901	60,0000	52,5198	27.05.2015 18:47:40	PASSED

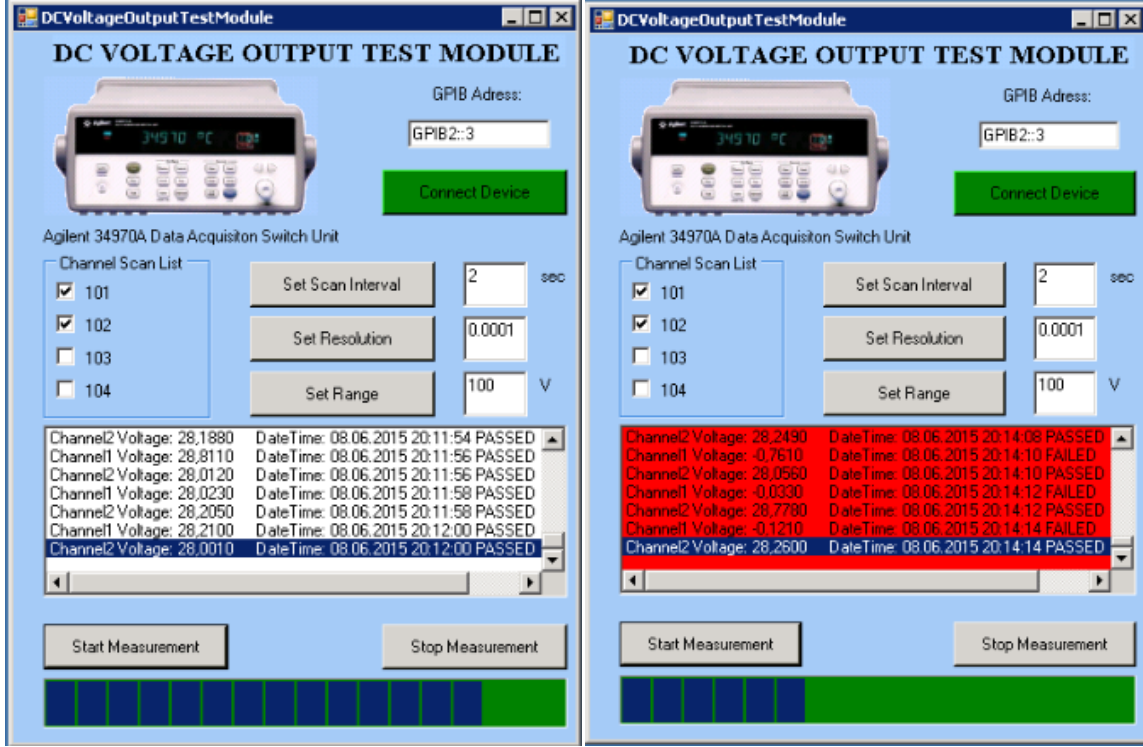
Şekil 25. AC Güç Kaynağı Test Sonuçları Ekran Görüntüsü

4.2.1.4 Güç Çıkış Arayüzü

TEY'in gösterge birimi alt sistemlerinin çalışması için gerekli DC besleme gerilimi, TEY'in içerisinde yer alan AC-DC güç kaynağı çeviricisi tarafından sağlanmaktadır. TEY'in sahip olduğu güç çıkış gerilimi arayüzü, gösterge birimi alt sisteminin düzgün olarak çalışabilmesi açısından önemlidir. Bu nedenle TEY'e yönelik gerçekleştirilen test doğrulama faaliyetleri sırasında DC güç çıkış geriliminin sürekli olarak ölçülmesi gerekmektedir.

İlgili testler sırasında Agilent firmasının 34970A Veri Toplama /Kaydetme Birimi kullanılmıştır [26]. Sistem testlerinin otomatize edilmesi amacıyla, TEY'in sahip olduğu güç çıkış arayüzü gerilim seviyelerinin testler süresinde sürekli olarak kontrol edilmesi ve test sonuçlarının kayıt altına alınması C# yazılım geliştirme ortamında DC gerilim çıkış test modülü simülatörü yazılarak sağlanmıştır.

İlgili test simülatörünün veri toplama/kaydetme birimi ile haberleşmesi sırasında Genel Amaçlı Arayüz Veri Yolu (GPIB), IEEE-488 haberleşme protokolü kullanılmıştır [31].



Şekil 26. DC Gerilim Çıkış Test Modülü Yazılımı Ekran Görüntüsü

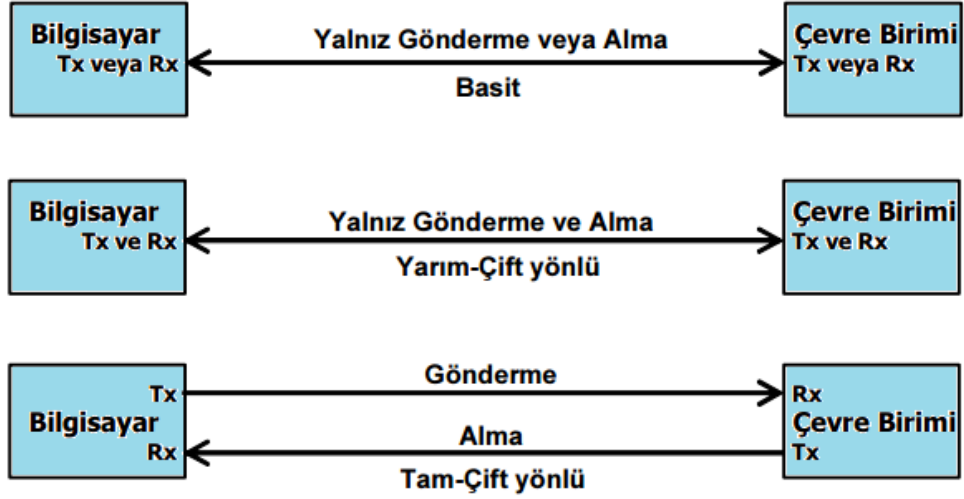
C# ortamında geliştirilen DC gerilim çıkış test modülü ekran görüntüleri Şekil 26’da sunulmaktadır. Şeklin sol tarafında otomatik olarak test edilen DC gerilim çıkış arayüzüne ait testin başarıyla tamamlandığı ekran görüntüsü yer alırken, sağ taraftaki görüntü otomatik yapılan testin başarısız sonuçlandığını göstermektedir. Diğer yandan, Şekil 27’de DC gerilim çıkış arayüzüne ait yapılan otomatik testin test sonuçlarının kayıt altına alındığını gösteren text (.txt) dosyasına ait ekran görüntüsü yer almaktadır.

```
DC_Voltage_Output_Logger.txt - Notepad
File Edit Format View Help
Channel1 voltage: 27,5380 DateTime: 08.06.2015 20:11:54 PASSED
Channel2 voltage: 28,1880 DateTime: 08.06.2015 20:11:54 PASSED
Channel1 voltage: 28,8110 DateTime: 08.06.2015 20:11:56 PASSED
Channel2 voltage: 28,0120 DateTime: 08.06.2015 20:11:56 PASSED
Channel1 voltage: 28,0230 DateTime: 08.06.2015 20:11:58 PASSED
Channel2 voltage: 28,2050 DateTime: 08.06.2015 20:11:58 PASSED
Channel1 voltage: 28,2100 DateTime: 08.06.2015 20:12:00 PASSED
Channel2 voltage: 28,0010 DateTime: 08.06.2015 20:12:00 PASSED
Channel1 voltage: 28,0230 DateTime: 08.06.2015 20:12:02 PASSED
Channel2 voltage: 28,0500 DateTime: 08.06.2015 20:12:02 PASSED
Channel1 voltage: 28,7730 DateTime: 08.06.2015 20:12:04 PASSED
Channel2 voltage: 28,0450 DateTime: 08.06.2015 20:12:04 PASSED
Channel1 voltage: 28,1990 DateTime: 08.06.2015 20:12:06 PASSED
Channel2 voltage: 28,2160 DateTime: 08.06.2015 20:12:06 PASSED
Channel1 voltage: -0,2210 DateTime: 08.06.2015 20:13:58 FAILED
Channel2 voltage: 28,2380 DateTime: 08.06.2015 20:13:58 PASSED
Channel1 voltage: 0,2920 DateTime: 08.06.2015 20:14:00 FAILED
Channel2 voltage: 28,1880 DateTime: 08.06.2015 20:14:00 PASSED
Channel1 voltage: -0,2650 DateTime: 08.06.2015 20:14:02 FAILED
Channel2 voltage: 28,7620 DateTime: 08.06.2015 20:14:02 PASSED
Channel1 voltage: -0,1760 DateTime: 08.06.2015 20:14:04 FAILED
Channel2 voltage: 28,7450 DateTime: 08.06.2015 20:14:04 PASSED
Channel1 voltage: -0,1710 DateTime: 08.06.2015 20:14:06 FAILED
Channel2 voltage: 27,5650 DateTime: 08.06.2015 20:14:06 PASSED
Channel1 voltage: -0,2540 DateTime: 08.06.2015 20:14:08 FAILED
Channel2 voltage: 28,2490 DateTime: 08.06.2015 20:14:08 PASSED
Channel1 voltage: -0,7610 DateTime: 08.06.2015 20:14:10 FAILED
Channel2 voltage: 28,0560 DateTime: 08.06.2015 20:14:10 PASSED
Channel1 voltage: -0,0330 DateTime: 08.06.2015 20:14:12 FAILED
Channel2 voltage: 28,7780 DateTime: 08.06.2015 20:14:12 PASSED
Channel1 voltage: -0,1210 DateTime: 08.06.2015 20:14:14 FAILED
```

Şekil 27. DC Gerilim Çıkış Test Sonuçları Ekran Görüntüsü

4.2.1.5 Seri Haberleşme Bağlantı Arayüzü

Seri haberleşme bağlantı arayüzü, terminaller ve çevre birimlerin birbiriyle olan bağlantısı sağlamak amacıyla kullanılan en genel haberleşme arayüzüdür (Bkz. Şekil 28). Seri haberleşme sırasında her seferinde gönderici ya da alıcı tarafında bir bit bilgi transfer edilmesi sağlanmakta olup, paralel haberleşmeye oranla sinyalin daha uzun mesafelere daha az kablo ile taşınabilmesine imkan sunmaktadır [31].



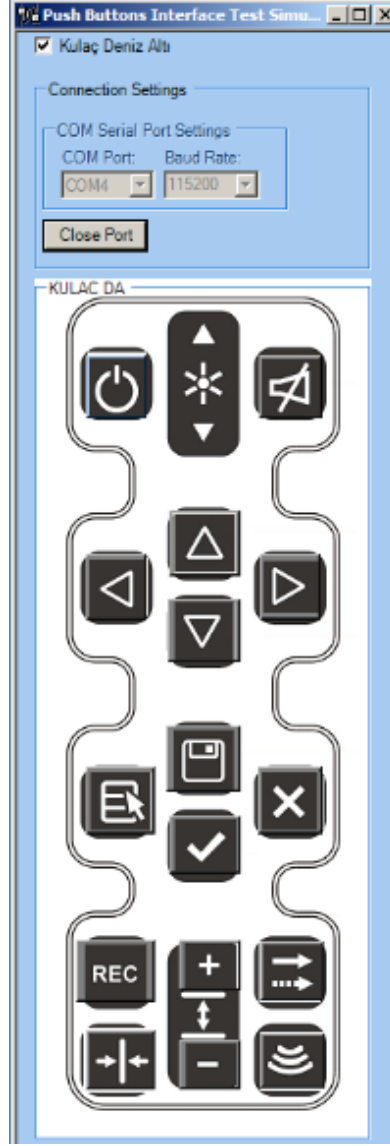
Şekil 28. Seri Kanal Haberleşme Bağlantı Arayüzü Gösterimi

Sistem testlerinin otomatize edilebilmesi için, TEY'in sahip olduğu seri haberleşme bağlantı arayüzü kullanılarak, sistemin genel işlevlerinin yerine getirilebilmesi amacıyla RS232/RS422 standartlarına uygun olarak simülatörlerin oluşturulması gerekmektedir.

- **Tuş Takımı Simülatörü**

Sistemin test senaryolarına uygun olarak, fonksiyonel testlerinin gerçekleştirilmesi amacıyla, TEY'in sahip olduğu tuş takımı panelinin yazılımsal olarak simüle edilmesi gerekmektedir.

Bu kapsamda seri kanal arayüzü (RS-232) kullanılarak sistemin tuş takımı paneli C# yazılım geliştirme ortamında yazılımsal olarak simüle edilmiş olup, ilgili simülatörüne yönelik ekran görüntüsü Şekil 29'da belirtilmiştir.

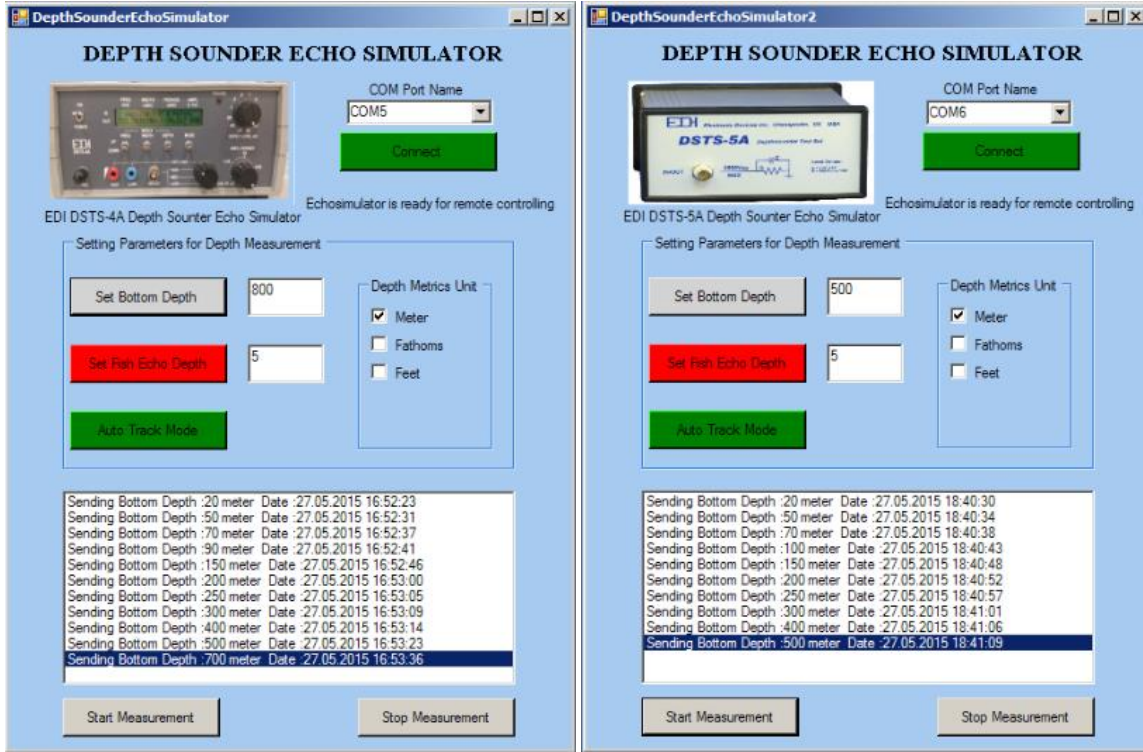


Şekil 29. Tuş Takımı Simülör Yazılımı Ekran Görüntüsü

- **Derinlik Simülörü**

Manuel test altyapısının oluşturulması sırasında kullanılan derinlik simülörleri 4.1.1.3 numaralı bölümde belirtilmişti. TEY için sistem testlerinin otomatize edilmesi aşamasında; insan hatalarının en aza indirgenmesi, test süresinden tasarruf edilmesi ve testlerin tekrarlanabilirliğinin artırılması için operatöre ihtiyaç duyulmaksızın sistemin başında bulunan bir test operatörü bulunmaması hedeflendiği için, ilgili derinlik simülörlerinin manuel olarak ayarlanabilmesi mümkün olmamaktadır.

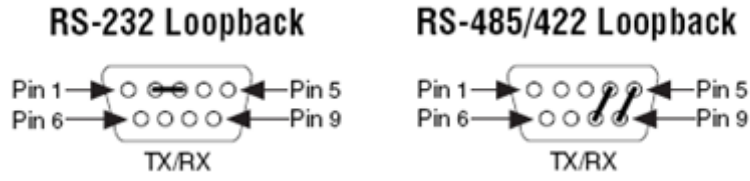
Bu bakımdan testlerin otomatizasyonu, ancak derinlik simülatörlerinin otomatik olarak kontrol edilebilmesi ile mümkün olmaktadır. Seri kanal haberleşme arayüzü kullanılarak sistemin derinlik simülatörleri, mesajlaşma formatlarına uygun olacak şekilde C# yazılım geliştirme ortamında yazılımsal olarak simüle edilmiş olup, ilgili simülatörlere yönelik ekran görüntüleri Şekil 30'da belirtilmiştir. Şeklin sol tarafında DSTS-4A düşük frekans sensör alt sistemini simüle eden derinlik simulatörü yer almaktadır. Şeklin sağ tarafında ise DSTS-5A yüksek frekans sensör alt sistemini simüle eden derinlik simulatörü yer almaktadır. Bu simulatörler aracılığı ile daha önce belirlenen test senaryosuna uygun olarak gönderilen derinlik verileri zamanla değiştirilerek testlerin senaryolara uygun olarak başarıyla gerçekleştirilmesi sağlanmaktadır.



Şekil 30. Derinlik Simülatörü Yazılımı Ekran Görüntüsü

- **Seri Kanal Haberleşme Arayüz Simülatörü (Loopback Testi)**

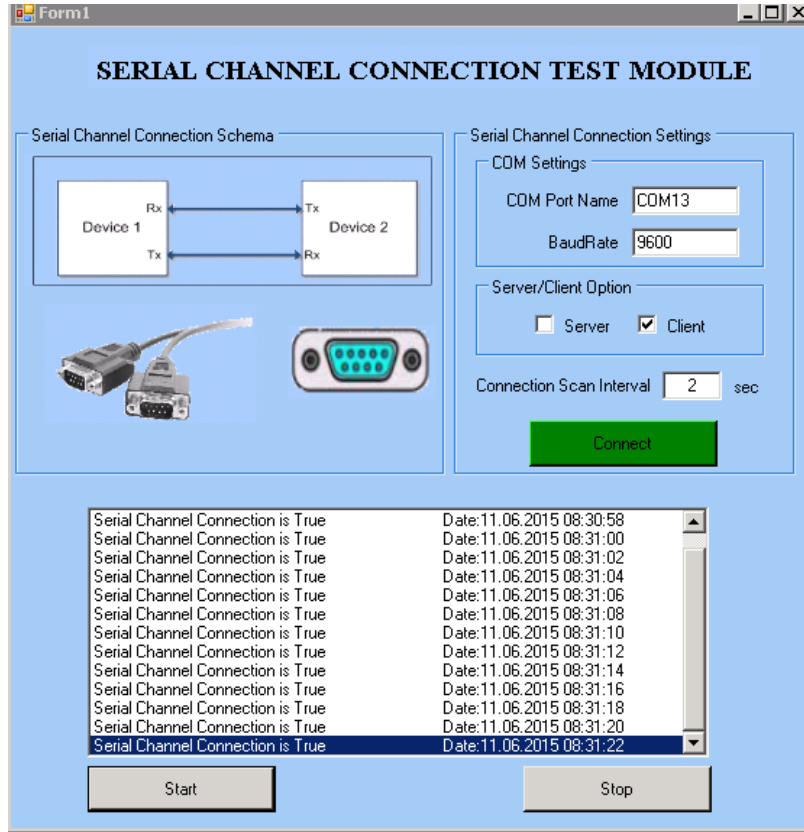
Geri dönüş testi; elektrik sinyallerinin haberleşme ağı içerisinde bulunan çevre birimlere gönderildiği sırada, gönderilen sinyallerin alıcı tarafına kısa devre bağlantısı sağlanarak sinyalin geri döndürüldüğü bir testtir. Geri dönüş testinin amacı, ilgili cihazın arayüz bağlantısının var olup olmadığını; bir başka deyişle çalışıp çalışmadığının kontrolü için gerçekleştirilmektedir. RS-232 ve RS-485/422 seri kanal arayüzleri için geri dönüş testi bağlantı gösterimi Şekil 31’de gösterilmektedir.



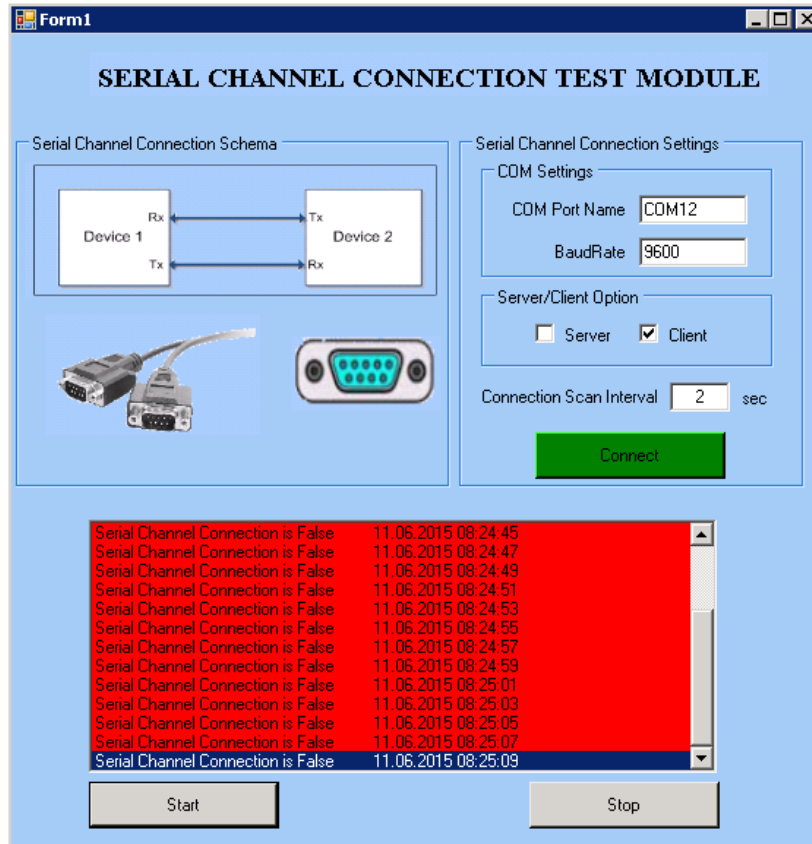
Şekil 31. Seri Kanal Loopback Testi Bağlantı Gösterimi

TEY’e yönelik sistem testlerinin otomatizasyonu için, TEY’in sahip olduğu yedek seri haberleşme bağlantı arayüzünün fiziksel olarak çalışabilirliğinin sürekli olarak kontrol edilmesi ve test sonuçlarının kayıt altına alınması C# yazılım geliştirme ortamında seri kanal bağlantı test modülü simülatörü yazılarak sağlanmıştır.

C# ortamında geliştirilen Seri kanal bağlantı test modülü ekran görüntüleri Şekil 32 ve Şekil 33’te sunulmaktadır. Şekil 32’de testin başarıyla tamamlandığı ekran görüntüsü yer alırken, Şekil 33’te ise testin başarısız sonuçlandığını ekran görüntüsü gösterilmektedir.



Şekil 32. Seri Kanal Bağlantı Test Modülü Yazılımı Ekran Görüntüsü (Başarılı)



Şekil 33. Seri Kanal Bağlantı Test Modülü Yazılımı Ekran Görüntüsü (Başarısız)

Sisteme ait fiziksel arayüz testlerinin otomatik olarak gerçekleştirilebilmesi amacıyla C# yazılım geliştirme ortamı kullanılarak yazılan test modülleri sayesinde:

- Donanımsal arayüz alternatifleri kapsanması
- Birden fazla arayüzün test edilmesine olanak sağlanması
- Standart haberleşme protokollerine uygun olarak testlerin gerçekleştirilmesi
- Çalışma sırasında gerçekleşen senaryoların kaydedilmesi ve raporlanması

işlemleri başarıyla gerçekleştirilmiştir.

4.2.2 Fonksiyonel Testlerin Otomatize Edilmesi

TEY'in Grafikselle Kullanıcı Arayüzüne ait bütün fonksiyonel testlerin senaryosal olarak gerçekleştirilmesi ve otomatize edilebilmesi için, Kaydet-Oynat yeteneğine sahip olan test yürütme araçlarının kullanılması gerekliliği ortaya çıkmıştır. Kaydet-Oynat yeteneğine sahip olan test yürütme araçları, kullanıcı arayüzünde yapılan davranışları kaydederek daha sonra sanki kullanıcı yapıyormuş gibi aynı hareketleri defalarca tekrarlayabilen araçlar olarak göze çarpmaktadır.

Birçok test otomasyon araçları, test senaryolarını oluşturmaları için kullanıcı eylemlerini kaydederek, sonrasında, istenilen zamanda istenilen sayıda onları geri kayıt ve oynatma özellikleri sağlar. Senaryolara uygun olarak gerçekleştirilen test faaliyetleri sonunda beklenen sonuçlarla gerçek sonuçları karşılaştırarak test başarımlarını otomatik olarak raporlanmaktadır.

Otomatik test yürütme araçlarının test doğrulama faaliyetleri sırasında kullanılması ile testlerin daha hızlı gerçekleştirilmesi sağlanabilmektedir [24] [29].

Bu kapsamda, piyasada bulunan otomatik test yürütme araçlarına yönelik araştırmalar gerçekleştirilmiş olup, piyasada bulunan belli başlı otomatik test yürütme araçlarına ilişkin karşılaştırmalı tablo detayı Çizelge 3'te belirtilmiştir [30].

Çizelge 3. Otomatik Test Yürütme Araçları Karşılaştırma Çizelgesi

	Testcomplete	Selenium	HP Quick Test Professional
Uygulama Desteği	Test complete için ayrıca eklenti satın almaya gerek yok. Hemen kurup test senaryoları tasarlanabilir.	Web tabanlı uygulamalar	Sadece müşteri sunucu uygulaması içerir. Ekstra eklentiler için ücret ödenmeli
Programlama Yeteneği	Kullanımı kolay	Programlama becerisi gerektirir.	Kullanımı kolay
İşletim Sistemi Desteği	Windows 7, Windows Vista, Windows Server 2008 ve sonrası	Windows PC/MAC/UNIXPlatforms.	Windows XP
Lisans Maliyeti	Lisans Gerekli (≈2300 \$)	Lisansa gerek yok. (Açık Kaynak)	Lisans Gerekli (≈8000 \$ Oldukça pahalı)
Desteklenen Test Tipleri	Regresyon Testi Birim Testi Fonksiyonel(GKA Testi) Web Servis Testi Kod Güdümlü Test	Regresyon Testi Birim Testi GKA Testi	Regresyon Testi Birim Testi Fonksiyonel(GKA Testi) Web Servis Testi
Nesne Tabanlı Dil Desteği	VBScript, JSScript, DelphiScript, C++Script ve C#Script,	Java, .Net, Perl, PHP, Python ve Ruby.	VBScript veya JavaScript.
Raporlama Özelliği	Test sonuçları kolaylıkla raporlanabiliyor.	Diğer otomatik test araçlarına göre iyi değil.	Test sonuçları kolaylıkla raporlanabiliyor.

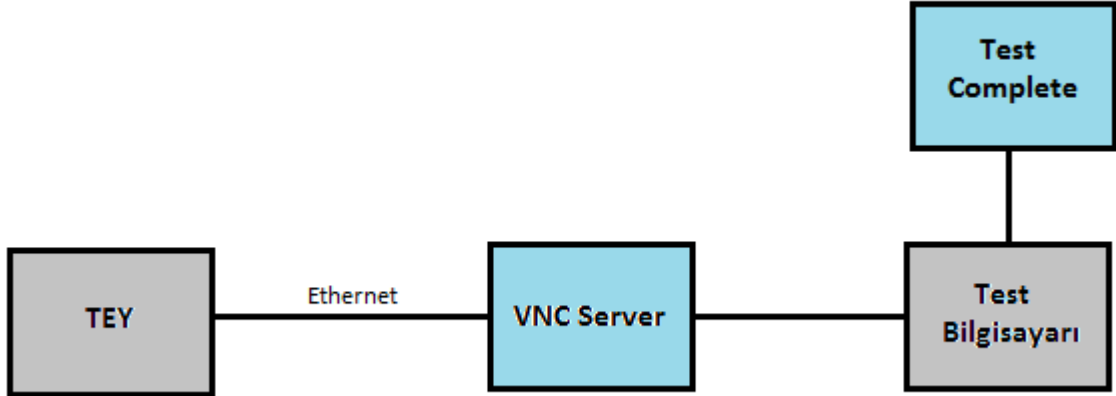
Bu bilgiler ışığında TEY'in Grafiksel Kullanıcı Arayüzüne ait bütün fonksiyonel testlerin senaryosal olarak gerçekleştirilmesi ve otomatize edilebilmesi için Testcomplete otomatik test yürütme aracının kullanılmasına karar verilmiş olup, Testcomplete test aracına yönelik bilgiler aşağıda özetlenmiştir.

- **Test Complete**

Testcomplete, otomatik test yürütme araçlarından birisi olarak birçok farklı yazılım test türlerini otomatik hale getirmek amacıyla kullanılmaktadır. Kayıt ve oynatma özelliği sebebiyle manuel testleri, test betiği kullanımına imkan vererek otomatik olarak testler tekrar tekrar oynatılabilir ve test betiği üzerinde yeni senaryo adımları eklenmesi ya da mevcut senaryo adımlarının üzerinde değişiklik yapılmasına izin verir.

Kaydedilen testler yeni testler oluşturmak veya mevcut testleri geliştirmek için test senaryoları daha sonra değiştirilebilmektedir.

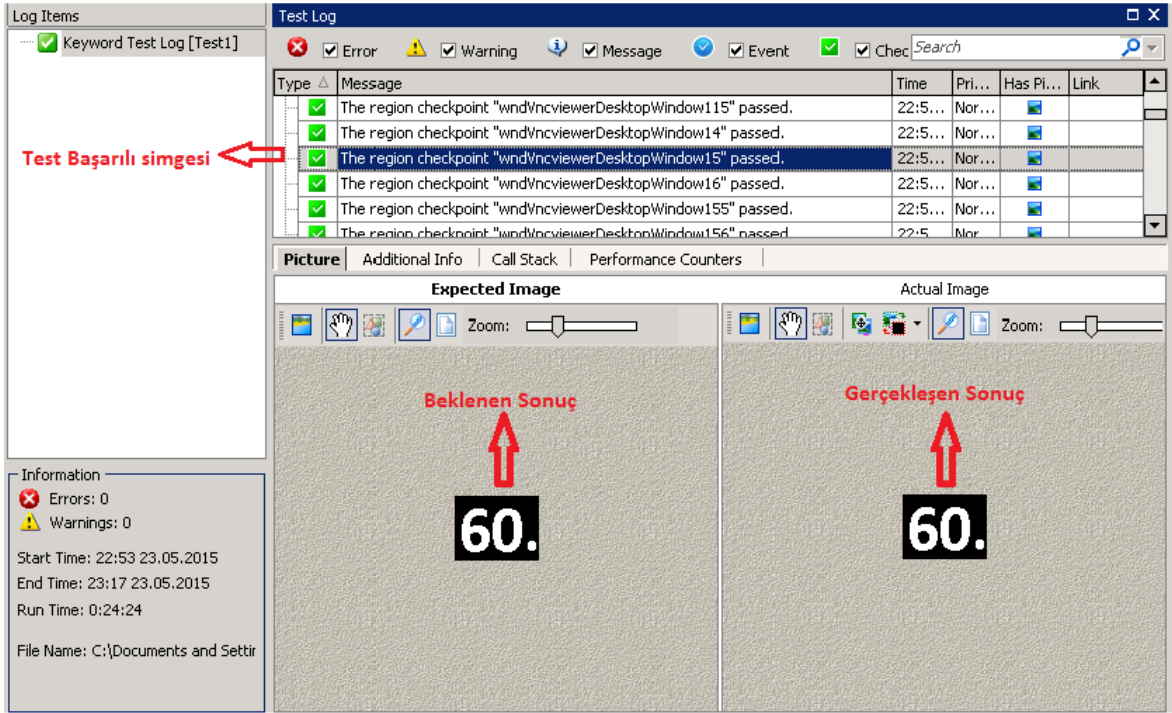
TEY'in GKA'sına yönelik fonksiyonel testlerin otomatize edilmesi için, TEY'in kullanıcı arayüzü görüntüsünün, ethernet arayüzü aracılığıyla üzerinde Testcomplete yürütme aracı bulunan test bilgisayarına aktarılması gerekmektedir. Bu aktarım işlemi için VNC Uzak kontrol yazılımı kullanılmakta olup, ilgili bağlantı detayı Şekil 34'te gösterilmiştir.



Şekil 34. TEY GKA Otomatizasyonu Bağlantı Şema Gösterimi

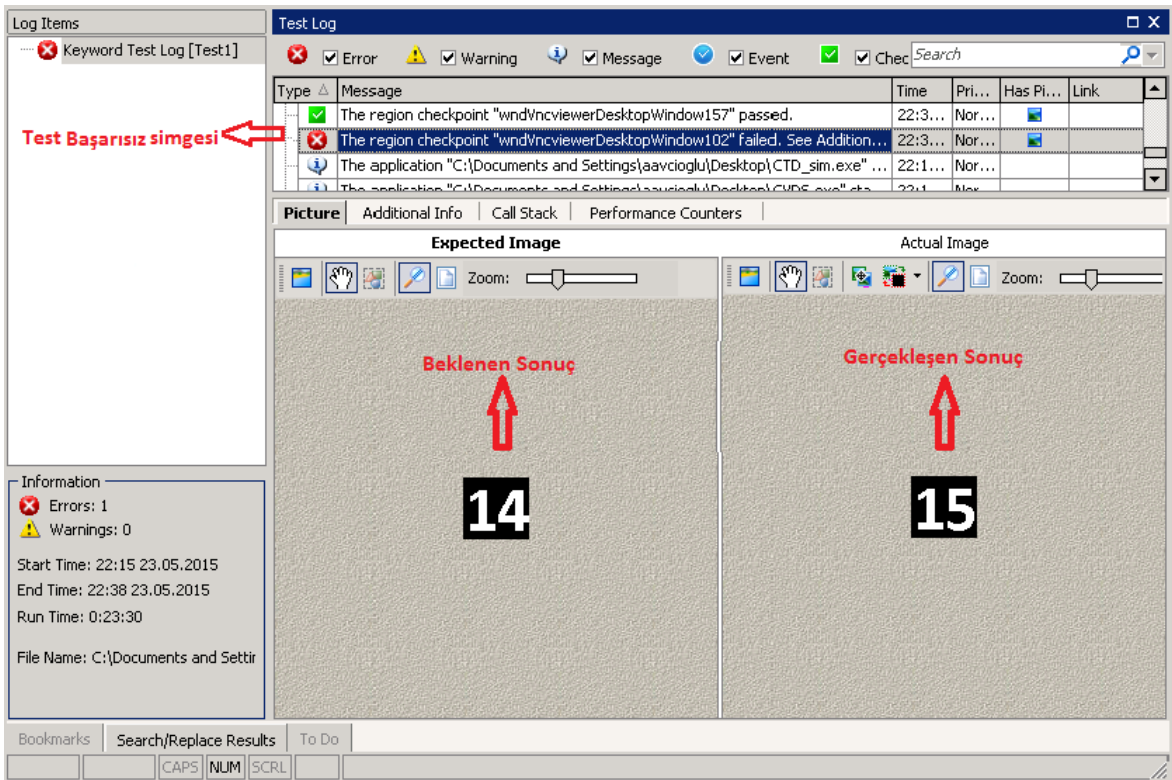
Oluşturulan altyapı ile birlikte TEY için kaydedilen senaryolar, Testcomplete yardımıyla, tekrar tekrar oynatılabilmektedir. Test faaliyetleri sonucunda test başarımları ise Testcomplete aracılığıyla beklenen sonuçlarla gerçek sonuçlar karşılaştırılarak, otomatik olarak oluşturulmaktadır.

Şekil 35'te kaydedilen senaryo adımına ilişkin beklenen sonuç ile gerçek sonucun karşılaştırılması ile başarılı olarak raporlanan ekran görüntüsü yer almaktadır.



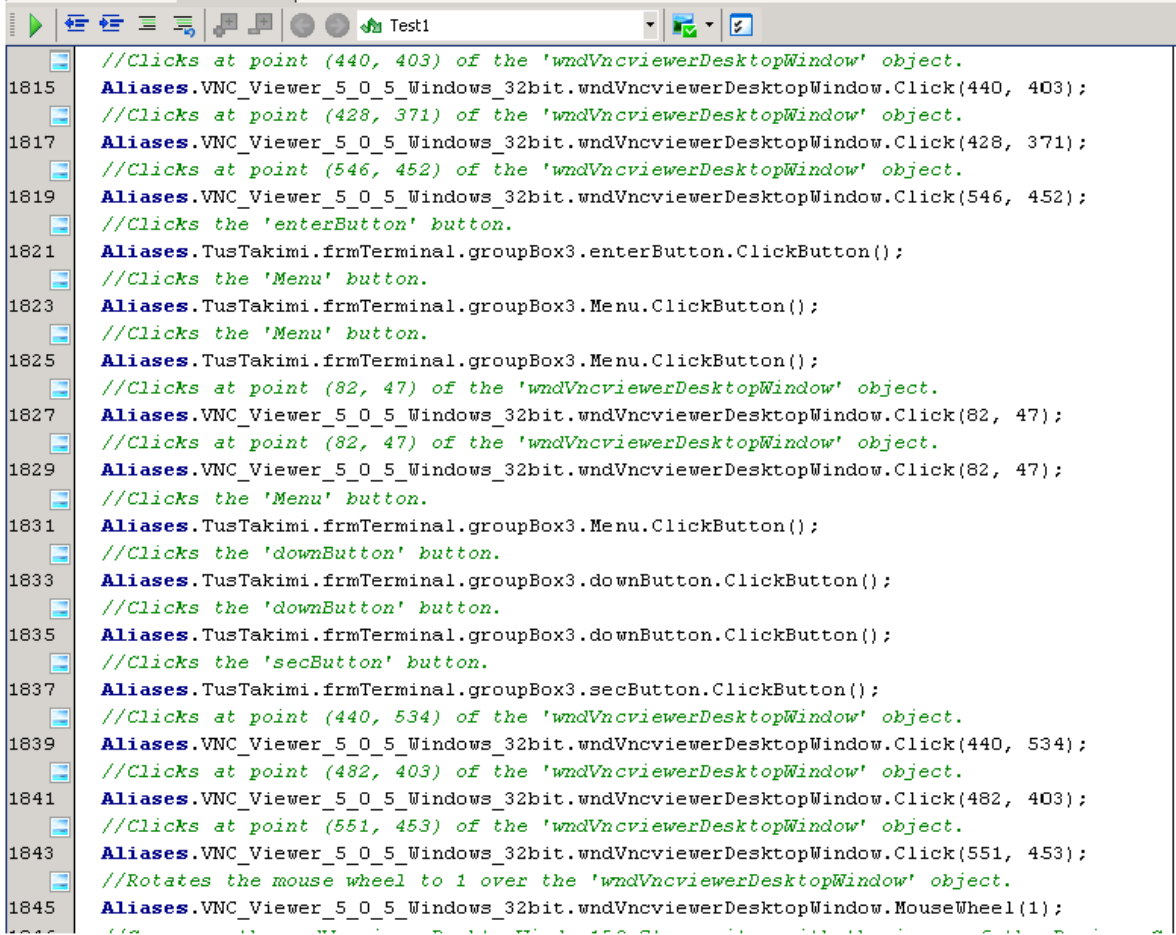
Şekil 35. Testcomplete Test Adımı Başarılı Gösterimi

Şekil 36'da ise kaydedilen senaryo adımına ilişkin beklenen sonuç ile gerçek sonucun karşılaştırılması ile başarısız olarak raporlanan ekran görüntüsü yer almaktadır.



Şekil 36. Testcomplete Test Adımı Başarısız Gösterimi

TEY'in GKA'sına ait bütün fonksiyonel testlerin senaryosal olarak gerçekleştirilmesi ve otomatize edilebilmesi için; her bir gereksinim maddesini doğrulamak amacıyla test senaryosu oluşturulmuş, Testcomplete yazılımı yardımıyla oluşturulan test senaryosu kayıt edilerek, test betiği haline çevrilmiştir. Oluşturulan Testcomplete test betiği örnek ekran görüntüsü Şekil 37'de görülmektedir.



```
1815 //Clicks at point (440, 403) of the 'wndVncviewerDesktopWindow' object.
Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(440, 403);
1817 //Clicks at point (428, 371) of the 'wndVncviewerDesktopWindow' object.
Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(428, 371);
1819 //Clicks at point (546, 452) of the 'wndVncviewerDesktopWindow' object.
Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(546, 452);
//Clicks the 'enterButton' button.
1821 Aliases.TusTakimi.frmTerminal.groupBox3.enterButton.ClickButton();
//Clicks the 'Menu' button.
1823 Aliases.TusTakimi.frmTerminal.groupBox3.Menu.ClickButton();
//Clicks the 'Menu' button.
1825 Aliases.TusTakimi.frmTerminal.groupBox3.Menu.ClickButton();
//Clicks at point (82, 47) of the 'wndVncviewerDesktopWindow' object.
1827 Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(82, 47);
//Clicks at point (82, 47) of the 'wndVncviewerDesktopWindow' object.
1829 Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(82, 47);
//Clicks the 'Menu' button.
1831 Aliases.TusTakimi.frmTerminal.groupBox3.Menu.ClickButton();
//Clicks the 'downButton' button.
1833 Aliases.TusTakimi.frmTerminal.groupBox3.downButton.ClickButton();
//Clicks the 'downButton' button.
1835 Aliases.TusTakimi.frmTerminal.groupBox3.downButton.ClickButton();
//Clicks the 'secButton' button.
1837 Aliases.TusTakimi.frmTerminal.groupBox3.secButton.ClickButton();
//Clicks at point (440, 534) of the 'wndVncviewerDesktopWindow' object.
1839 Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(440, 534);
//Clicks at point (482, 403) of the 'wndVncviewerDesktopWindow' object.
1841 Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(482, 403);
//Clicks at point (551, 453) of the 'wndVncviewerDesktopWindow' object.
1843 Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.Click(551, 453);
//Rotates the mouse wheel to 1 over the 'wndVncviewerDesktopWindow' object.
1845 Aliases.VNC_Viewer_5_0_5_Windows_32bit.wndVncviewerDesktopWindow.MouseWheel(1);
```

Şekil 37. Testcomplete Test Betiği Kesit Gösterimi

Testcomplete yazılımı yardımıyla kaydedilen test senaryosu uygun biçimde yürütülerek, test faaliyetleri sonucunda test başarımları durumu, beklenen sonuçlarla gerçek sonuçların karşılaştırılması yoluyla otomatik olarak oluşmaktadır. Kayıt edilen test senaryosuna yönelik test sonuç ekranı Şekil 38'de gösterilmektedir.

Log Items

Test Log

Keyword Test Log [Test1]

Test Sonucu Başarılı
(Tüm Senaryo için Test Sonucu)

Toplam Test Süresi

Information

Errors: 0

Warnings: 0

Start Time: 22:53 23.05.2015

End Time: 23:17 23.05.2015

Run Time: 0:24:24

File Name: C:\Documents and Settir

Test Log

Error Warning Message Event Checkpoint

Type	Message	Time
✓	The region checkpoint "wndVncviewerDesktopWindow1" passed.	22:54:36
✓	The region checkpoint "wndVncviewerDesktopWindow105" passed.	22:54:39
✓	The region checkpoint "wndVncviewerDesktopWindow106" passed.	22:54:40
✓	The region checkpoint "wndVncviewerDesktopWindow107" passed.	22:54:40
✓	The region checkpoint "wndVncviewerDesktopWindow108" passed.	22:54:41
✓	The region checkpoint "wndVncviewerDesktopWindow2" passed.	22:54:42
✓	The region checkpoint "wndVncviewerDesktopWindow3" passed.	22:54:42
✓	The region checkpoint "wndVncviewerDesktopWindow4" passed.	22:54:43
✓	The region checkpoint "wndVncviewerDesktopWindow5" passed.	22:54:44
✓	The region checkpoint "wndVncviewerDesktopWindow6" passed.	22:54:44
✓	The region checkpoint "wndVncviewerDesktopWindow7" passed.	22:54:45
✓	The region checkpoint "wndVncviewerDesktopWindow8" passed.	22:54:46
✓	The region checkpoint "wndVncviewerDesktopWindow9" passed.	22:54:46
✓	The region checkpoint "wndVncviewerDesktopWindow10" passed.	22:54:47
✓	The region checkpoint "wndVncviewerDesktopWindow11" passed.	22:54:48
✓	The region checkpoint "wndVncviewerDesktopWindow12" passed.	22:54:48
✓	The region checkpoint "wndVncviewerDesktopWindow13" passed.	22:54:58
✓	The region checkpoint "wndVncviewerDesktopWindow109" passed.	22:55:05
✓	The region checkpoint "wndVncviewerDesktopWindow110" passed.	22:55:09
✓	The region checkpoint "wndVncviewerDesktopWindow111" passed.	22:55:16
✓	The region checkpoint "wndVncviewerDesktopWindow112" passed.	22:55:22
✓	The region checkpoint "wndVncviewerDesktopWindow113" passed.	22:55:27
✓	The region checkpoint "wndVncviewerDesktopWindow114" passed.	22:55:36
✓	The region checkpoint "wndVncviewerDesktopWindow115" passed.	22:55:40
✓	The region checkpoint "wndVncviewerDesktopWindow14" passed.	22:55:49
✓	The region checkpoint "wndVncviewerDesktopWindow15" passed.	22:56:40
✓	The region checkpoint "wndVncviewerDesktopWindow16" passed.	22:56:41
✓	The region checkpoint "wndVncviewerDesktopWindow155" passed.	22:56:53
✓	The region checkpoint "wndVncviewerDesktopWindow156" passed.	22:57:22
✓	The region checkpoint "wndVncviewerDesktopWindow17" passed.	22:57:45
✓	The region checkpoint "wndVncviewerDesktopWindow18" passed.	22:57:54
✓	The region checkpoint "wndVncviewerDesktopWindow19" passed.	22:58:09
✓	The region checkpoint "wndVncviewerDesktopWindow20" passed.	22:58:10
✓	The region checkpoint "wndVncviewerDesktopWindow21" passed.	22:58:16
✓	The region checkpoint "wndVncviewerDesktopWindow22" passed.	22:58:16
✓	The region checkpoint "wndVncviewerDesktopWindow23" passed.	22:58:37
✓	The region checkpoint "wndVncviewerDesktopWindow24" passed.	22:58:38
✓	The region checkpoint "wndVncviewerDesktopWindow25" passed.	22:58:52

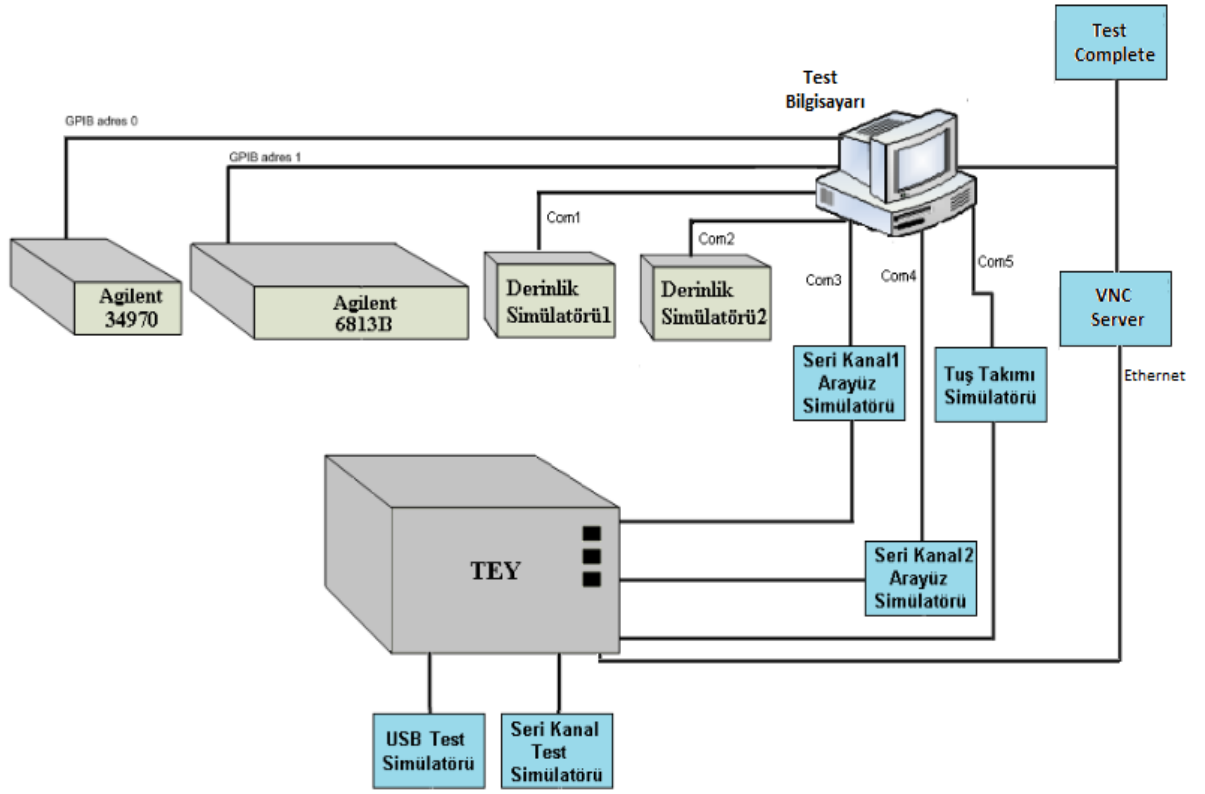
Picture Additional Info Call Stack Performance Counters

Expected Image Actual Image

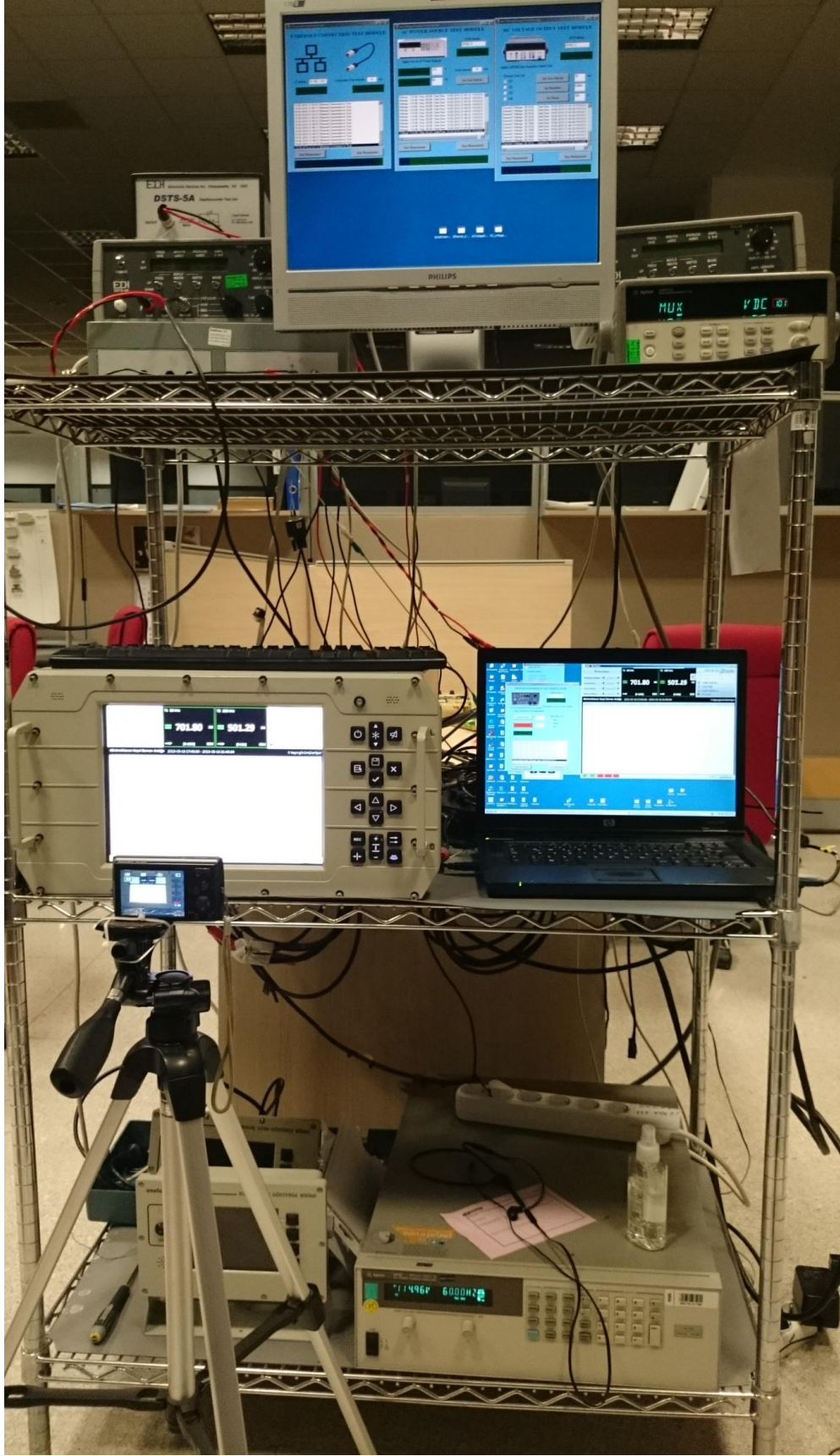
100 100

Şekil 38. Testcomplete Test Sonuç Gösterimi

Bu çalışmalara istinaden, TEY'e yönelik sistem testlerinin otomatizasyonu için, fiziksel arayüz testlerinin ve grafiksel kullanıcı arayüzü fonksiyonel testlerinin yürütülmesi için geliştirilen otomatize test altyapısı Şekil 39 ve 40'da gösterilmektedir.



Şekil 39. Otomatize Test Altyapısı Blok Gösterimi



Şekil 40. Otomatize Test Altyapısı

5. SONUÇLAR

Tez kapsamında, çok sayıda platform arayüzünden oluşan bilgisayar tabanlı KULAÇ sisteminin sistem (yazılım/donanım) test faaliyetlerinin otomatik hale getirilmesine yönelik test altyapısı geliştirme çalışmaları anlatılmıştır.

KULAÇ Derinlik Ölçüm Sonarı Sistemi; bir su üstü veya su altı platformuna entegre olan, dalış derinliğini ve/veya deniz dibi derinliğini ölçen sistemdir. Sisteme ait sistem testleri normal şartlar altında gemi platformlarında gerçek donanımlarla gerçekleştirilmektedir. Bu çalışma hem maliyetli hem de verimsiz olduğundan, sistem testlerinin deniz ortamından önce entegrasyon ortamında doğrulanması ve üretim sürecinde sistem testlerinin laboratuvar ortamında gerçekleştirilmesi amacıyla çevre birimler/alt sistemler manuel test altyapısı ile simüle edilmiştir. Geliştirilen manuel test altyapısı ile KULAÇ sistemine ait sistem testleri laboratuvar ortamında operatör yardımı ile gerçekleştirilmiştir.

Sonraki adımda; manuel test altyapısı ile sağlanan simüle deniz ortamında KULAÇ sisteminde, operatöre ihtiyaç duyulmaksızın, tasarlanan ve geliştirilen test altyapısı ile sistem testleri (fiziksel arayüz testleri ve grafiksel kullanıcı arayüz testleri) otomatik olarak gerçekleştirilmiştir.

Manuel ve otomatize test altyapıları geliştirme çalışmaları kapsamında iki test altyapısı ile ilgili karşılaştırma aşağıda özetlenmiştir:

Çizelge 4. Test Altyapıları Karşılaştırma Çizelgesi

	Test Altyapısı Geliştirme Süresi	Test Yürütme Süresi**
Manuel Test* (Test Operatörü)	450 saat	≈ 1 saat 30 dakika
Otomatik Test	800 (450+350) saat	≈ 25 dakika

(*) Test faaliyetlerini gerçekleştiren test operatörü süresi 1 adet operatör iş gücü baz alınarak ölçülmüştür.

(**) Test yürütme süreleri 1 döngüde gerçekleştirilen test süresi olarak hesaplanmış olup, çevre koşulları için gerçekleştirilen işlevsel testler sırasında 4 döngülük testler gerçekleştirilmektedir.

Oluşturulan test altyapısı ile KULAÇ Sistemi test faaliyetleri (sistem testleri, üretim testleri, regresyon testleri), otomatize edilmiş olup, bu sayede;

- Test faaliyetleri esnasında İnsan kaynaklı hatalar en aza indirgenmiştir.
- Testler aralıksız 7x24 kořturulabilmiştir.
- Testler daha sık tekrarlanabilir hale gelmiştir.
- Test süreleri, test esnasında ihtiyaç duyulan insan gücü dolayısıyla iş yükü büyük ölçüde azalmış olup, verimlilik sağlanmıştır.

Bu çalışma, geliştirilen otomatize test altyapısı ile ASELSAN'da geliştirilmekte olan sistemlerin test faaliyetlerine yönelik sistematik bir yaklaşım getirmiştir. KULAÇ sisteminde olduğu gibi birçok arayüze sahip sistemlerin sistem test faaliyetleri maliyetli olduğundan, geliştirilen otomatize test altyapısı ile sistem testi maliyetleri düşürülmüş, zamandan tasarruf edilmiş ve sistem testleri daha sağlıklı ve tekrarlanabilir olarak yapılmaya başlanmıştır.

Geliştirilen otomatize test altyapısı modüler ve ölçeklenebilir olduğundan; gelişen teknoloji ile gelecekte tasarlanacak daha karmaşık sistemlerin testleri için de temel oluşturacak olup, değişik projelerde ihtiyaçlara yönelik ilave modüller geliştirilerek daha kompleks sistemlerin sistem testleri gerçekleştirilebilecektir. Ayrıca değişik projelerde ihtiyaç duyulacak test altyapılarının daha kısa sürede hazırlanmasına ve kullanabilmesine imkan tanıyacaktır.

KAYNAKLAR

- [1] D. A. U. Press, *System Engineering Fundamentals*, Fort Belvoir, **2001**.
- [2] J.N. Martin, *Systems Engineering Guidebook: A Process for Developing Systems and Products*, Lucent Technologies, **1997**.
- [3] NASA Systems Engineering Handbook, Revision 1, NASA/SP-2007-6105, NASA, **2007**.
- [4] A. Sato, M. Miki, T. Yamanouchi and M. Watanabe, *Software Synthesis For Trade-off Design*, Syracuse (USA), **2002**.
- [5] A. Felix, *Standard Approach to Trade Studies: A Process Improvement Model that Enables Systems Engineers to Provide Information to the Project Manager by Going Beyond the Summary Matrix*, Navair (CA), **2004**.
- [6] D. W. Hubbard, *The Failure of Risk Management: Why It's Broken and How to Fix It*, John Wiley and Sons, **2009**.
- [7] Forsberg, Kevin; Mooz, Harold, The Relationship of System Engineering to the Project Cycle, *In Proceedings of the First Annual Symposium of National Council on System Engineering*, **1991**.
- [8] A. Ghahrai, V Model, <http://www.testingexcellence.com/v-model> (Haziran, **2015**)
- [9] M. Palmieri, *System Testing in a Simulated Environment*, **2013**.
- [10] P.Hut, CMM and Project Quality Management, <http://www.pmhut.com/cmm-and-project-quality-management> (Haziran, **2015**)
- [11] J. Elfriede, G. Thom and G. Bernie, *Implementing Automated Software Testing: How to Save Time and Lower Costs While Raising Quality*, Addison Wesley, **2009**.
- [12] A. Engel, *Verification, Validation and Testing of Engineered Systems*, John Wiley & Sons, **2010**.
- [13] N. R. Pusuluri, *Software Testing Concepts And Tools*, Dreamtech Press, **2006**.
- [14] M. Fowler, Refactoring, <http://martinfowler.com/refactoring>, (Haziran, **2015**)
- [15] S. Elbaum, G. Erothermel and J. Penix, *Techniques For Improving Regression Testing In Continuous Integration Development Environments*, FSE, **2014**.
- [16] K. Beck, *Test-driven development: by example*, Addison-Wesley Professional, **2003**.
- [17] Microsoft, Integration Testing, <https://msdn.microsoft.com/en-us/library/aa292128>, (Haziran, **2015**)
- [18] IEEE, *A Compilation of IEEE Standard Computer Glossaries*, New York, NY, **1991**.
- [19] Hardware-in-the-loop simulation, https://en.wikipedia.org/wiki/Hardware-in-the-loop_simulation, (Haziran, **2015**)

- [20] A. Avizienis, J.-C. Laprie, B. Randell and C. Landwehr, in Basic Concepts and Taxonomy of Dependable and Secure Computing, vol. I, IEEE, **2004**.
- [21] Hardware-in-the-loop simulation, https://en.wikipedia.org/wiki/Hardware-in-the-loop_simulation, (Haziran, **2015**)
- [22] M. Alam, *Software Test Automation Myths and Facts*, Dallas, **2010**.
- [23] B. Pettichord, *Seven Steps to Test Automation Success*, San Jose (USA), **2001**.
- [24] E. Alegroth, R. Feldt, and H. Olsson, Transitioning Manual System Test Suites to Automated Testing: An Industrial Case Study, *ICST*, **2012**.
- [25] Underwater acoustics, https://en.wikipedia.org/wiki/Underwater_acoustics, (Haziran, **2015**)
- [26] 34970A Data Acquisition / Data Logger Switch Unit, <http://www.keysight.com/en/pd-1000001313:epsg:pro-pn-34970A/data-acquisition-data-logger-switch-unit?&cc=TR&lc=eng>, (Haziran, **2015**)
- [27] User's Guide AC Power Solutions Agilent Models 6811B, 6812B, and 6813B, Agilent Technologies, **2004**.
- [28] Depth Sounder Test Set, EDI, <http://www.dsts.com/index.htm>, (Haziran, **2015**)
- [29] E. Borjesson and R. Feldt, Automated System Testing using Visual GUI Testing Tools: A Comparative Study in Industry, *ICST*, **2012**.
- [30] H.Kaur, G.Gupta, Comparative Study of Automated Testing Tools: Selenium, Quick Test Professional and Testcomplete, *IJERA*, **2013**.
- [31] B. Güven,T.Uzun, Kontrol Sistemlerinde Kullanılan Veri Haberleşmesi Teknolojileri,http://www.habtekus.yildiz.edu.tr/2007/cd/bildiriler/haberlesme_uygulamalari/39.pdf, (Haziran, **2015**)
- [32] Sonar, <https://en.wikipedia.org/wiki/Sonar>, (Haziran, **2015**)
- [33] Kulaç Derinlik Ölçme Sonar Sistemi (Iskandil), <http://www.c4defence.com/wp-content/uploads/2015/03/KULA%C3%87.pdf>, (Haziran, **2015**)
- [34] Derinlik Ölçümü İçin Akustik "KULAÇ", http://www.aselsan.com.tr/tr-tr/basin-odasi/Documents/ASELSANDergileri/86.2012_3.pdf, (Haziran, **2015**)
- [35] L. Luo, *Software Testing Techniques*, Institute for Software Research International Carnegie Mellon University, Pittsburgh (USA), **2001**.

ÖZGEÇMİŞ

Kimlik Bilgileri

Adı Soyadı : Alper AVCIOĞLU
Doğum Yeri : AFYON
Medeni Hali : Bekar
E-posta : avcioglualper@gmail.com
Adresi : Mehmet Akif Ersoy Mah. 296. Cad. NO:16, ASELSAN A.Ş.
06370, Yenimahalle/ANKARA/TÜRKİYE

Eğitim

Lise : Afyon Süleyman Demirel Fen Lisesi
Lisans : İstanbul Teknik Üniversitesi

Yabancı Dil ve Düzeyi

İngilizce : İyi düzeyde

İş Deneyimi

ASELSAN A.Ş Sistem Test Tasarım Mühendisliği (2011)

Deneyim Alanları

Otomatize Test Sistemleri, Su Altı Akustik Sistemleri

Tezden Üretilmiş Projeler ve Bütçesi

-

Tezden Üretilmiş Yayınlar

-

Tezden Üretilmiş Tebliğ ve/veya Poster Sunumu ile Katıldığı Toplantılar

-