

**FIRAT ÜNİVERSİTESİ**

**TEKNOLOJİ FAKÜLTESİ**

**Yazılım Mühendisliği**

****

**YMH319 – Programlama Dilleri Dersi**

**Proje Uygulaması ve Dokümantasyonu**

**TEZ DOKÜMAN KONTROL SİSTEMİ**

*185541013* Abdullah AYGÜN

**Ocak – 2021**

1. GİRİŞ

# ÖNSÖZ

Tez yazma, bilimsel araştırma sürecinin son aşamasıdır. Tez metni yazımındaki temel amaç, araştırma problemi ile ilgili bulguları, yorumları, genellemeleri ve önceden yapılmış olan çalışmaları okuyucuya ulaştırmaktır.

Tez metni biçimsel ve içeriksel olarak belli standartları karşılamalıdır. İçerik yönünden açık ve anlaşılır bir dille yazılması gereken tez metni, biçimsel olarak da pek çok standart kurala uymak zorundadır.

Akademik camiada, bu biçimsel kurallar neredeyse evrensel nitelik kazanmıştır. Bu çerçevede üniversitemiz bünyesinde de tez yazım kurallarının standardını bildiren bir kılavuz hazırlanmıştır.

Proje kapsamında, bu kılavuz esas alınarak, yazılmış olan bir tez metninin biçimsel yönden standartlara uyup uymadığının tespiti amaçlanmaktadır.

Proje çıktısı sayesinde kullanıcı, hazırlamış olduğu tez metnini biçimsel yönden denetleyerek çalışmasının, tez yazım kurallarına ne derece uygun olduğunu görecek ve gereken düzeltmeleri gerçekleştirebilecektir.

# Projenin Amacı

Proje, bir tez dokümanı yazarken yapılan biçimsel hataları minimuma indirmek ve bu hataları kullanıcılara göstererek düzeltilmelerini sağlamayı hedefler. Böylece, kullanıcı, hazırlamış olduğu tez çalışmasını hatasız bir şekilde gerekli makama teslim edebilecektir.

Tez yazım kontrolü için aşağıdaki özel hedeflere yönelik bir yazılım geliştirilmesine ihtiyaç olduğu ortaya çıkmıştır:

* Sayfa kenar boşluklarının istenen değerlerde olup olmamasının,
* Antetlerin doğru bir şekilde konumlandırılıp konumlandırılmadığının,
* Tez başlığının büyüklüğü, hizası ve yazı tipinin istenilen değerlerde olup olmadığının,
* Verilen isimlerin doğru yazılıp yazılmadığının,
* Sayfa numaralarının hizasının ve doğruluğunun,
* Her bir satır aralığının istenen değer olup olmadığının,
* İçindekiler bölümündeki sayfa numaralarıyla tez sayfalarının aynı olup olmadığının,
* Ana bölüm başlıklarının kurallara uygun olup olmadığının,
* Paragraf düzeninin istenen değerler olup olmadığının,
* Türkçe özet ve Abstract kısımlarındaki anahtar kelime sayılarının istenen sayılarda olup olmadığının,
* Numaralandırma kurallarına uyup uymadığını,
* Başlık içeriklerinin satır aralıkları, hizalaması ve yazı tipinin istenen değerlerde olup olmamasını,
* Denklemlerin paragraf girintisi yapıp yapmadığının kontrolünün sağlanmasıdır.

# Projenin Kapsamı

Her türlü lisans ve yüksek lisans yapan öğrencilerin tez yazarken kullanabilecekleri yazılımlar kapsamındadır. Öte yandan, öğrencilerin hazırlamış olduğu bu tezlerin kontrolünü sağlamak amacıyla tezin teslim edileceği kişiler de kullanabilecektir.

Proje, herhangi bir tez dokümanının OpenXML kütüphanelerini kullanarak tez çalışması yapan lisans ve yüksek lisans öğrenciler tarafından ihtiyaç duyulan programın Pascal (Delphi) ile kodlanmasını kapsamakta olup; istenen girdiler ile maksimum hataları en aza indirgemektedir.

# Tanımlamalar ve Kısaltmalar

Punto: Tez içerisinde kullanılan harflerin boyutunu ifade eder.

Font: Yazı tipidir.

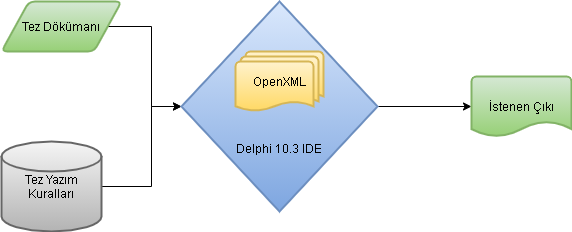
TCS: Thesis Control System, genel anlamı dışında proje çıktısı olan yazılımımızı tanımlamaktadır.

XML: Extensible Markup Language (Genişletilebilir İşaretleme Dili) hem insanlar hem bilgi işlem sistemleri tarafından kolayca okunabilecek dokümanlar oluşturmaya yarayan bir işaretleme dilidir.

2. PROJE PLANI

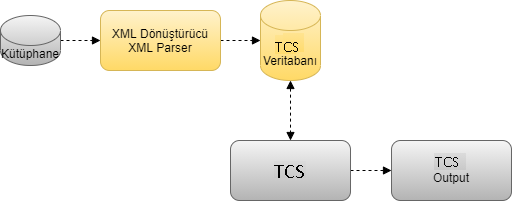
# 2.1 Giriş

Günümüz lisans ve lisansüstü eğitimleri için hazırlanan tez çalışmaları son derece önem arz etmektedir. Kişi, öğrenim hayatı boyunca edindiği tecrübeleri bir proje çıktısı şeklinde alır ve aldığı bu çıktıları belli bir formattaki dokümana kayıt eder.



Şekil 1 TCS çalışma şekli

# 2.2 Projenin Plan Kapsamı

Tez yazarken hem içerik olarak mantıklı yazılmalı hem de sözdizimi kurallarına uyulması gerekmektedir. Bu nedenle proje çıktısı sayesinde sözdizimi hatalarının belli bir kısmı önlenebilmektedir.

Şekil 2 Projenin Genel Yapısı

##### AYARLANMAMIŞ İŞLEV NOKTASI PUANININ (AİP) BULUNMASI

AİP = 1 \* W(1) + 2 \* W(2) + 3 \* W(3) + 4 \* W(4) + 5 \* W(5)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Ölçüm Parametresi | Ağırlık Faktörü (W) | | | | | | | Parametre Değeri | |
| **Sayı** | **Yalın** | | **Ortalama** | | **Karmaşık** | |
| (1)Kullanıcı Girdi sayısı | - | | 3 | | 4 | | 6 | |  |
| (2)Kullanıcı Çıktı sayısı | 8 | | 4 | | 5 | | 7 | | 40 |
| (3)Kullanıcı Sorgu Sayısı | - | | 3 | | 4 | | 6 | |  |
| (4)Kütük Sayısı | 4 | | 7 | | 10 | | 15 | | 28 |
| (5)Dışsal Arayüz Sayısı | - | | 5 | | 7 | | 10 | |  |
| TOPLAM |  | |  | |  | |  | | **68** |

##### TEKNİK KARMAŞIKLIK FAKTÖRÜNÜN (TKF) BULUNMASI

|  |  |  |
| --- | --- | --- |
| 1. | Sistem güvenilir yedekleme ve kurtarma gerektiriyor mu? | 0 |
| 2. | Veri iletişimi gerekiyor mu? | 3 |
| 3. | Dağıtık fonksiyon var mı? | 2 |
| 4. | Performans kritik mi? | 5 |
| 5. | Sistem çok kullanılan bir işletim ortamında mı çalışacak? | 2 |
| 6. | Sistem on-line veri girişi gerektiriyor mu? | 0 |
| 7. | On-line veri giriş işlemlerinin birden fazla ekran ya da işlem üzerinden olmasını mı gerektiriyor? | 0 |
| 8. | Ana dosyalar on-line mı güncelleniyor? | 0 |
| 9. | Girdiler, çıktılar, dosyalar ve sorgular karmaşık mı? | 5 |
| 10. | Kod yeniden kullanabilir olarak mı tasarlanmış? | 5 |
| 11. | İç süreç karmaşık mı? | 4 |
| 12. | Dönüşüm ve kurulum tasarım içerisinde mi? | 1 |
| 13. | Uygulama değişik kuruluşlarda birden fazla kurulum gerektirecek şekilde mi tasarlanmış? | 0 |
| 14. | Uygulama kullanıcı tarafından kolaylıkla kullanmayı ve değiştirmek üzere mi tasarlanmış? | 5 |
|  | **0:** Hiçbir Etkisi Yok **1:** Çok Az etkisi var **2**: Etkisi Var **3**: Ortalama Etkisi Var  **4**: Önemli Etkisi Var **5**: Mutlaka Olmalı, Kaçınılamaz | 32 |

##### İŞLEV NOKTASI PUANININ (UFP) BULUNMASI

İN = AİN x (0.65 x 0.01 x TKF)

İN= 68\*(0,65\*0,01\*32)

İN = **14,144**

##### SATIR SAYISININ BULUNMASI

Proje kodlamasında kullanılacak olan dil C# olup, satır sayısı kestirim katsayısı 60 dur.

Satır Sayısı = İN \* 60 = 15,028 \* 60 **= 901**

##### ETKİN MALİYET MODELİ: COCOMO

Projemiz, küçük ölçekli bir proje olduğundan Temel Model baz alınacaktır. Modelin uygulamasında kullanılan formüller aşağıdaki gibidir:

|  |  |  |
| --- | --- | --- |
| Ayrık Projeler | Yarı Gömülü Projeler | Gömülü Projeler |
| İş Gücü K = 2.4\*S1,05  Zaman T = 2.5\*K0,38 | İş Gücü K = 3,0\*S1,12  Zaman T = 2.5\*K0,35 | İş Gücü K = 3,6\*S1,20  Zaman T = 2.5\*K0,32 |
| S: 1000 türünden satır sayısı | | |

“Ayrık Proje” sınıfına giren projemiz için kullanacağımız COCOMO Formülü şu şekilde ortaya çıkmaktadır:

İş Gücü K = 2.4 \* S1,05 = 2.4 \* 0.51,05 = 1.16 (Aylık Kişi Başı İş Gücü)

Zaman T = 2.5 \* K0,38 = 2.5 \* 1.160.38= 2.9 = 3 Ay

# 2.3 Proje Ekip Yapısı

Şekil 3Proje Ekibi ve Görevleri

# 2.4 Proje Zaman-İş Planı

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ZAMAN  İŞ | Proje Başlangıç Tarihi:06/10/2020 | | | |  |  | | | | | | | | HAFTALAR | | | | | | |
| 1 | 2 | 3 | | | | 4 | 5 | | 6 | 7 | 8 | | | 9 | 10 | | 11 | 12 |
| İhtiyaç / Gereksinim Yönetimi, Analizi |  |  |  | | | |  |  | |  |  |  | | |  |  | |  |  |
| İhtiyaç / Gereksinim Geliştirme |  |  |  | | | |  |  | |  |  |  | | |  |  | |  |  |
| Sistem Çözümleme |  |  |  | | | |  |  | |  |  |  | | |  |  | |  |  |
| Kullanıcı Arayüz Tasarımı |  |  |  | | | |  |  | |  |  |  | | |  |  | |  |  |
| Gerçekleştirim |  |  |  | | | |  |  | |  |  |  | | |  |  | |  |  |
| Sistem Doğrulanması |  |  |  | | | |  |  | |  |  |  | | |  |  | |  |  |
| Organizasyonel Eğitimler |  |  |  | | | |  |  | |  |  |  | | |  |  | |  |  |
|  | Abdullah AYGÜN | | | Abdullah AYGÜN | | | | | Abdullah AYGÜN | | | | Abdullah AYGÜN | | | | Abdullah AYGÜN | | | |

Tablo 1Proje İş-Zaman Çizelgesi

# 2.5 Önerilen Sistemin Teknik Tanımları

* Herhangi bir tez doküman çalışmasını tabular forma çeviren bir kütüphane parser modülü bulunmaktadır.
* Parse edilmiş XML formatındaki dosyadan alınan veriler ile kural tabanlı çalışması sağlanmıştır.
* Bu tez yazım kuralları ile belirlenen bir formatta tez çalışması oluşturulması sağlanmıştır.
* Elde edilmiş çıktının sunumu zengin ve gelişmiş kullanıcı dostu UI ( User Interface) özellikleri sunmaktadır.
* Program bağlılığı (dependency) minimum seviyede tutulmuş olup, stand-alone bir şekilde çalışabilmektedir.

# 2.6 Kullanılan Özel Geliştirme Araçları ve Ortamları

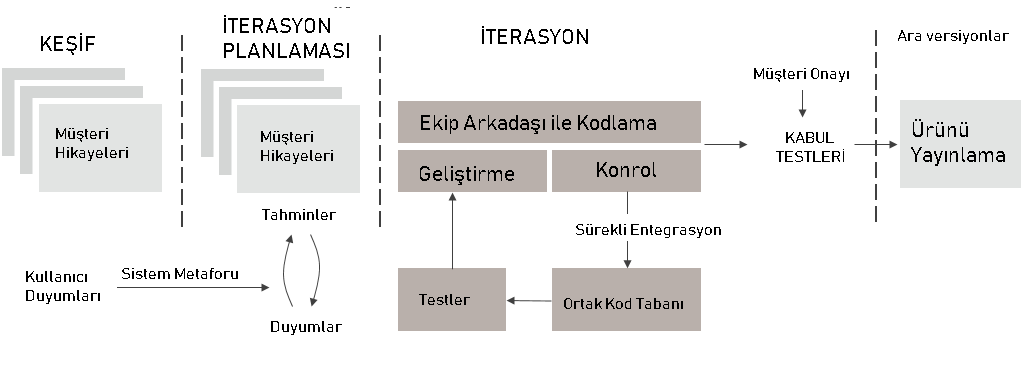
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | ÇÖZÜMLEME  TASARIM | PROGRAMLAMA | SINAMA | DESTEK |
| ORTAMLAR | Windows 10 | Delphi 10.3 IDE | Delphi 10.3 IDE |  |
| ARAÇLAR | Microsoft Word  Microsoft Visio | DevExpress, OpenXML |  | Google Chrome |

# 2.7 Proje Standartları, Yöntem ve Metodolojiler

Projemizin, 12 hafta gibi kısa sürede tamamlanması öngörülmüştür. TCS ‘nin robust yapısı ve yazılacak olan modüllerin konsept olarak ilişkili olsalar bile kodlama açısından her birinin bağımsız küçük iş parçacıkları olması ve bunların küçük iterasyonlarla kodlanıp, üzerinde paydaşların görüş alış-verişi yapacağı toplantılarla oluşturulacak olması, doğal olarak Agile Metodolojisi’nin yaygın olarak kullanılacağına işaret etmektedir.

Bir Agile uygulaması olarak SCRUM, her ne kadar popülaritesi olan bir metodoloji olsa da sertifikalı bir SCRUM Master’ın varlığını gerektirdiğinden projemizde uygulanamayacaktır.

Extreme Programming temelinde bulunan Basitlik, Cesaret, Geri Dönüş, İletişim kullanacağımız Agile metodolojisi için projemizle uygunluk arz etmektedir. Özellikle bu 4 temel maddenin en önemli maddeleri olan Geri Dönüş ve Ekibin birbirleri ile iletişiminin sağlıklı tutulması projemizin ekip yapısına uymaktadır. Geri Dönüş yani İstekler, proje yapım aşamasındayken bile değişebilir ve bu değişim diğer yazılım metodolojilerine göre daha hızlı uygulanabilir.



Şekil 4 Extreme Programlama Metodolojisi Akışı

# 2.8 Kalite Sağlama Planı

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| İş Tanımı | Zaman | | | | | | | | | | | | | İlgili Kişi | | Kalite Sağlama Unsuru | | Kullanılacak Kaynak | |
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |  | |  | |  | |
| Planlama |  |  |  |  |  |  |  |  |  |  |  |  | Abdullah  AYGÜN | | Projenin tüm gereksinimlerini karşılayacak şekilde yürütülmesi için gereken kalite standartlarının değerlendirilmesi | | IEEE 730 Standard for  Software Quality Assurance Plans,  Üniversitemiz Bölüm Yetkilileri | |
| Ürün gözden geçirme |  |  | | | | | | | | |  |  | Abdullah  AYGÜN | | Etkinlik  Verimlilik  Müşteri Memnuniyeti | | JSTD-  016 Standart for Information Technology Software Life  Cycle Processes Software Development | |
|  | | | | | | | | |
| Süreç/proje takip |  |  | | | | | | | | |  |  | Abdullah AYGÜN | | Tanımlama soruları  Uygulama soruları  İzleme  soruları | | ISO/IEC29500  standartı | |
|  | | | | | | | | |
| Konfigürasyon tetkikleri |  |  |  |  |  |  |  |  |  |  |  |  | Abdullah AYGÜN | | İşlevsel Özellikler  Performans  Özellikleri  Dokümantasyon  Uyumu | | WordprocessingML Reference | |

# 2.9 Konfigürasyon Yönetim Planı

|  |  |  |  |
| --- | --- | --- | --- |
| Konfigüre Gerektiren Durum | İlgili Kişi | Süre | Sonuç |
| OpenXML SDK ihtiyacı | Abdullah AYGÜN | 1 gün | Veri işleme UI komponentlerinin kurulum ve konfigürasyonu yapılacaktır. |
| Yeni kural tanımlama | Abdullah AYGÜN | 1 gün | TCS ’nın import edeceği dosyaya ait konfigürasyon yapılacaktır. |

# 2.10 Kaynak Yönetim Planı

|  |  |  |
| --- | --- | --- |
| Kaynak Türü | Proje Kapsamında Elverişlilik | Kullanım Amacı |
| İnsan | 1 Öğrenci – Programcı | Proje Ekibi |
| Malzeme |  |  |
| Araç, gereç, donanım | 1. laptop bilgisayar | Projenin kodlanması |
| İşletme Sermayesi | - | - |

Sistemin gerçekleştirilmesi için gereken donanım mevcut olup, herhangi bir maliyete katlanmadan projenin gerçekleştirilmesini mümkün kılacaktır.

OpenXML SDK ‘nın ücretsiz lisansı (Trial Version), yazılıma herhangi bir ücret ödemesi gerektirmediğinden, projede sermayeye ihtiyaç duyulmamıştır.

Yazılım Mühendisliği bölümü 1 öğrencinin ders projesi olarak gerçekleştireceği yazılım için, herhangi bir ücret ödemesi olmayacaktır.

# 2.11 Eğitim Planı

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Eğitim Türü | Eğitim Konusu | Eğitim Verecek Kişi | Tarih/Süre | Amaç |
| OnLine | TCS Kullanımı | Abdullah AYGÜN | 08/01/2021 | Kullanıcı arayüzünün tanıtılması |

# 2.12 Test Planı

Projemizde yazılım test faaliyetleri süresince birim, entegrasyon ve sistem testleri gerçekleştirilecektir. Aşağıdaki yazılım modülleri bu plan kapsamında belirtilen testlere tabi tutulacaktır.

Projemizde yazılım test faaliyetleri süresince birim, entegrasyon ve sistem testleri gerçekleştirilecektir. Aşağıdaki yazılım modülleri bu plan kapsamında belirtilen testlere tabi tutulacaktır.

* *TCS🡪docx modülü*

Import edilen kütüphane modülü istenildiği şekilde mesaj döndürüyor mu?

* *Kontrol modülü*

Kontrol, amaçlandığı TCS kabiliyetlerine uygun çalışıyor mu?

* *Görünüm menüsü*

Görsellik, istenilen düzeyde mi?

İstenilen şekilde özelleştirme yapılabiliniyor mu?

1. Derece Hatalar: Sistemin genel çalışmasını direkt etkileyen hatalar.
2. Derece Hatalar: Sistemin genel çalışmasını direkt etkilemeyen fakat işlevsel olarak bazı kısımların çalışmasını engelleyen hatalar.
3. Derece Hatalar: Sistemin çalışmasını engellemeyen, görsel yönden ortaya çıkan hatalardır.

Proje kapsamında test seviyelerine göre görev dağılımı ve sorumluluklar şöyledir:

|  |  |
| --- | --- |
| Test Seviyeleri | Görev ve Sorumluluklar |
| Birim Testleri | Geliştirme Ekibi |
| Entegrasyon Testleri | Test Ekibi – Abdullah AYGÜN |
| Sistem Testleri | Test Ekibi – Abdullah AYGÜN |
| Kabul Testleri | Müşteri ve Test Ekibi - Abdullah AYGÜN |

Projesi kapsamında test faaliyetleri yazılımcı laptop bilgisayarı üzerinde yürütülecektir. Bilgisayara ait özellikler aşağıdaki gibidir:

CPU: Intel Xeon 3.3 GHz

RAM: 8 GB DDR4 2133 MHz

HDD: 100 GB

OS: Windows 10

Yüklenecek Programlar: Microsoft Excel

|  |  |
| --- | --- |
| Test Araçları | Kullanım Amacı |
| Firebase Test Lab | Birim testleri için geliştirme ekibi tarafından |
| Winium | Sistem testleri için test ekibi tarafından test otomasyon aracı kullanılacaktır. |
| Apache JMeter | Fonksiyonel olmayan testler (Performans, Yük ve Stres) için Apache JMeter aracı kullanılacaktır. |
| TFS | Test faaliyetleri sonucunda bulunan hatalar TFS üzerinde ilgili geliştiriciye hata kaydı olarak açılacaktır. |

# 2.13 Bakım Planı

Proje ile gerçekleştirilen yazılım, standalone bir sistem olduğu için, periyodik sistem yedeklemesi de dahil bir bakım gerektirmeyecektir. Müşteri, istediği output dosyasını analiz edip elde ettiği tüm içeriklerin kopyasını alabilir, bu ilk ana output dosyasında bir değişiklik gerektirmez.

3. SİSTEM ÇÖZÜMLEME

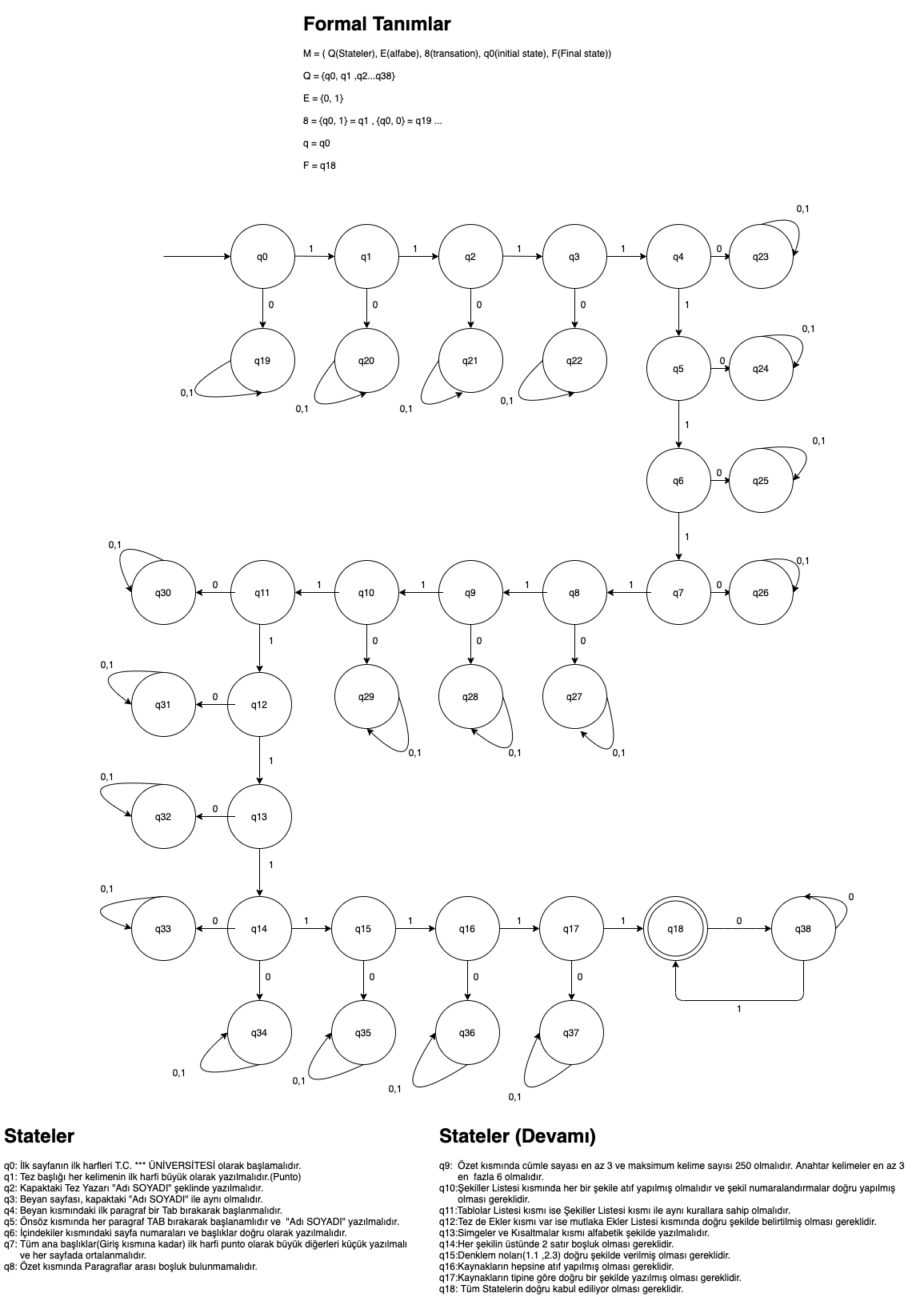
# 3.1 Mevcut Sistem İncelemesi

Proje ile standart bir tez şablonundan farklı bir işleyiş şekli tasarlanmıştır. Normalde müşteri ya da kullanıcı bir şablona göre yazdığı dokümanda yaptığı hataları kendisi inceleyerek fark edebilir ve düzeltebilir. Ancak TCS sayesinde kural yapısından geçirilen bir dokümandan kısa bir sürede hata çıktıları almak mümkündür.

## Örgüt Yapısı

Şekil 5 TCS Organizasyonal Yapısı

## İşlevsel Model



## Veri Modeli

TCS ‘ye kaynak teşkil eden veri yapısı tez kuralları şablonu ile export edilen bir tez dokümanın içerisindeki hataları yüksek hassasiyetle bulmaya çalışır. Bulduğu hatalar kullanıcıya çıkı olarak verilir.

## Var olan Yazılım/Donanım Kaynakları

Tez Kontrol Sistemleri programları açık kaynak, ücretsiz yazılımlar olup, her türlü platformda sorunsuz bir şekilde çalışabilme özelliklerine sahiptir. Ancak programların sağladığı zenginleştirilmiş ve geliştirilmiş filtreleme özelliklerini taşımamaktadır. Söz konusu yazılımlar minimum konfigürasyona sahip bir PC’de kolaylıkla çalıştırılabilir. Dolayısıyla geliştirmiş olduğumuz ürün de kendi başına, minimum konfigürasyon ve gereksinimli bir PCde çalışabilecektir.

## Varolan Sistemin Değerlendirilmesi

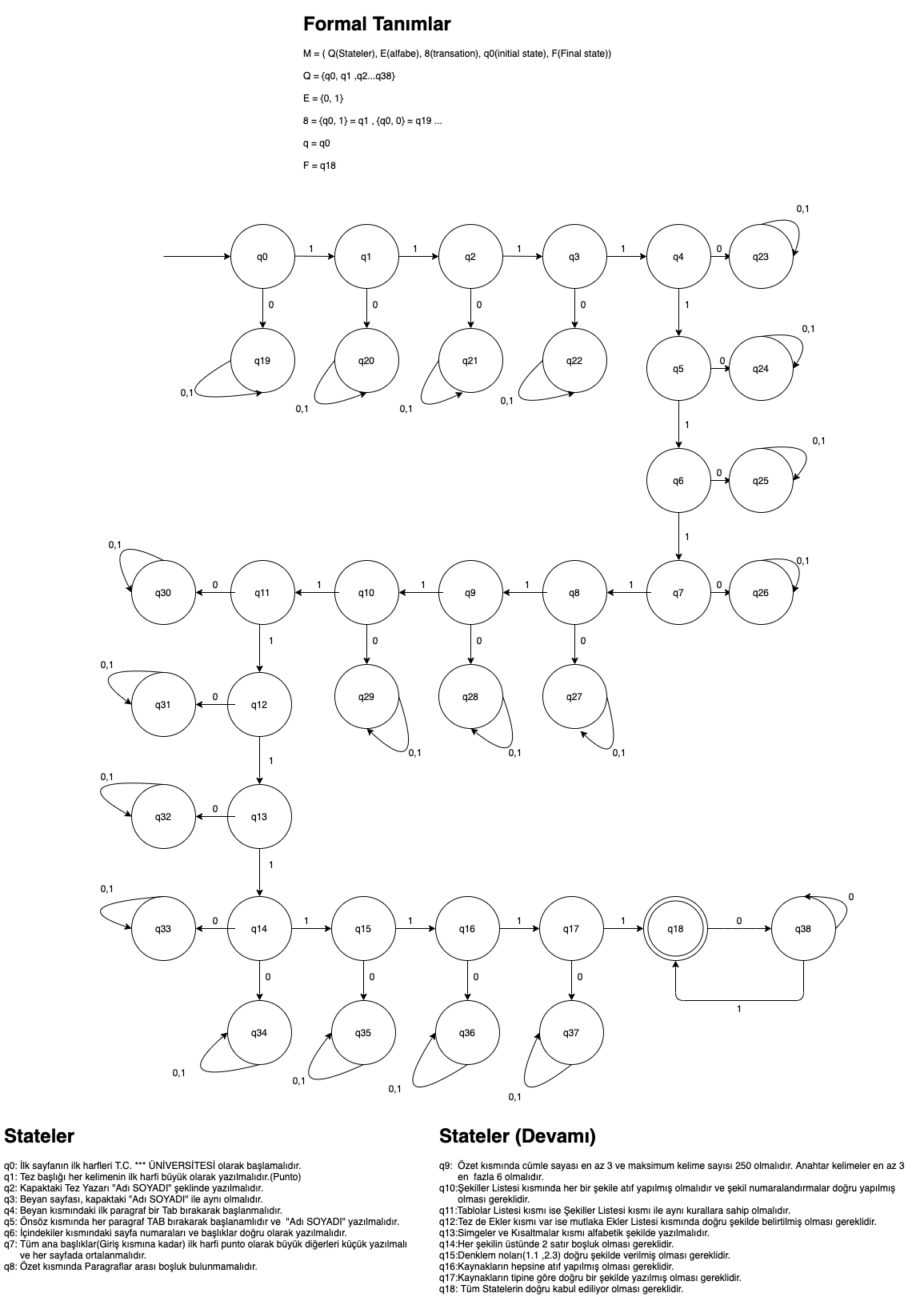
Üniversitemize ait belirlenen bir şablon olduğundan dolayı başka bir kural yapısı ile karşılaşıldığı zaman program hatalı çalışacaktır. Bu nedenle piyasada tez kontrol sistemi yazılımları nadir derecede bulunmaktadır. Bu programlar, yapılmış olan projedeki program ile benzer özellikler göstermekle beraber, yapılmış olan proje çıktı programı başta güzel bir kullanıcı arayüzü ile kendi kurallarını kendisi tanıtmaktadır.

# 3.2 Gereksenen Sistemin Mantıksal Modeli

## 3.2.1 Giriş

Projenin daha da kullanıcı dostu çalışabilmesi adına ayrı bir platformda verilen çıktılar Tez dokümanında yorum satırı olarak gösterilebilir. Böylece daha kullanışlı ve native çalışacağından performanslı olur. Bu sebeple TCS, Delphi dili ile geliştirilmiştir.

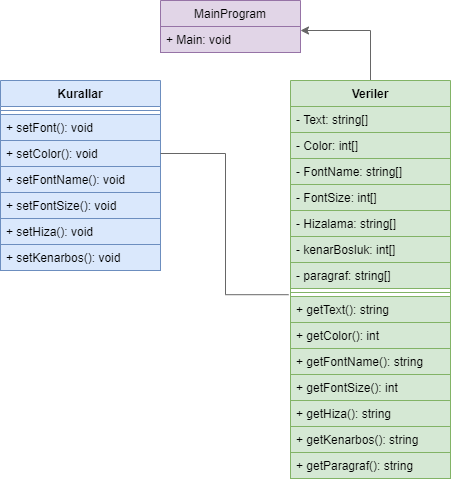
## 3.2.2 İşlevsel Model



## 3.2.3 Genel Bakış

Proje kapsamında söz konusu tek bir dokümanın analizi gerekecektir. Bu kapsamda projeye özel karmaşık bir sınıf-nesne yapısı yoktur.

## 3.2.4 Bilgi Sistemleri/Nesneler

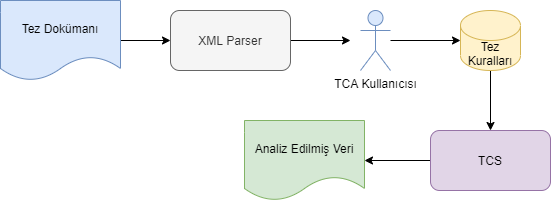


Şekil 8 Proje Sınıf-Nesne İlişkileri

## 3.2.5 Veri Sözlüğü

Programın kullanılmasına dair özel bir veri sözlüğü bulunmamakla birlikte, ürün tipik bir veritabanı kullanıcısının rahatlıkla adapte olabileceği bir arayüze sahiptir.

## 3.2.6 İşlevlerin Sıradüzeni



Şekil 10 İşlevlerin Sırası

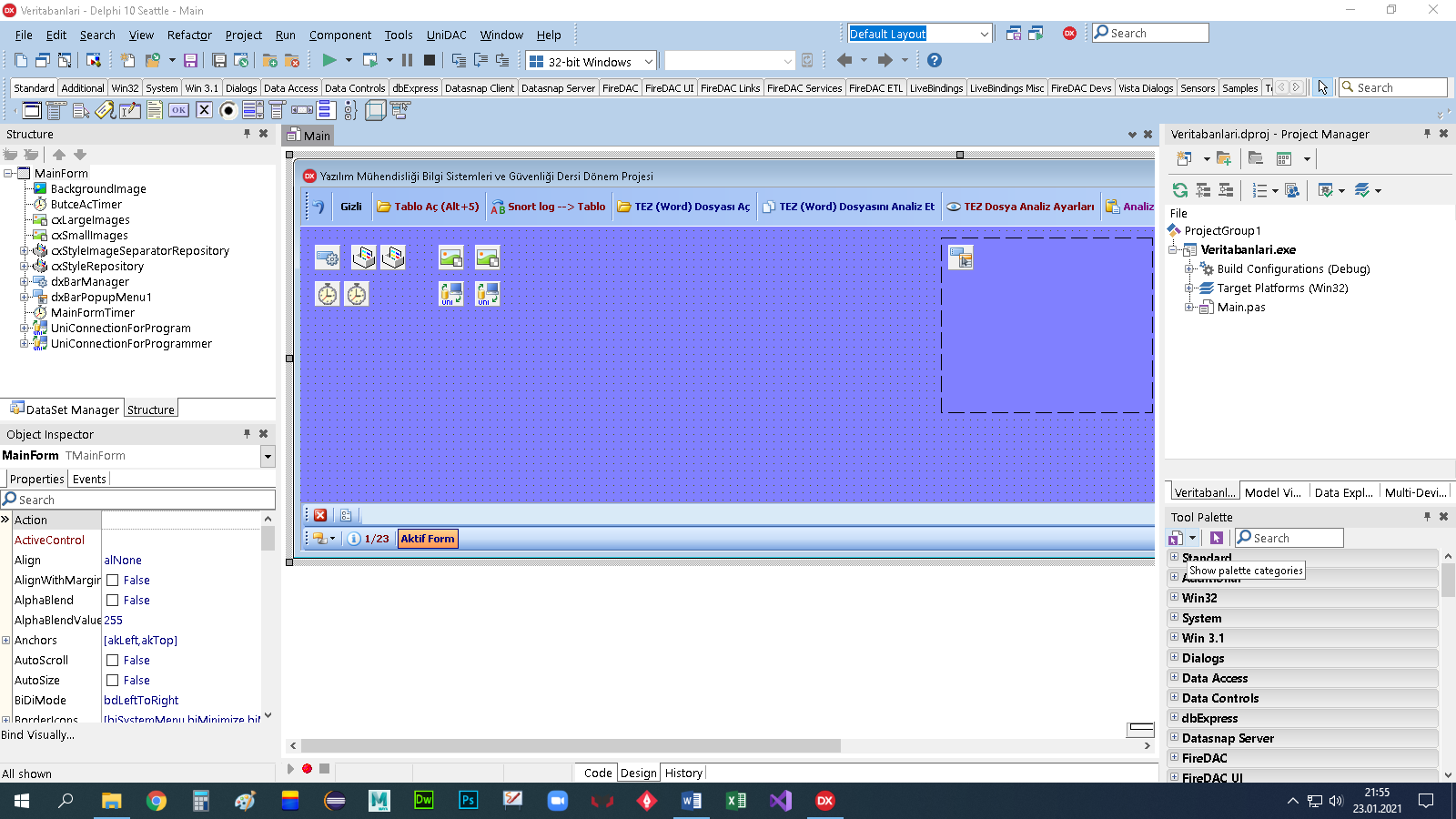
## 3.2.7 Başarım Gerekleri

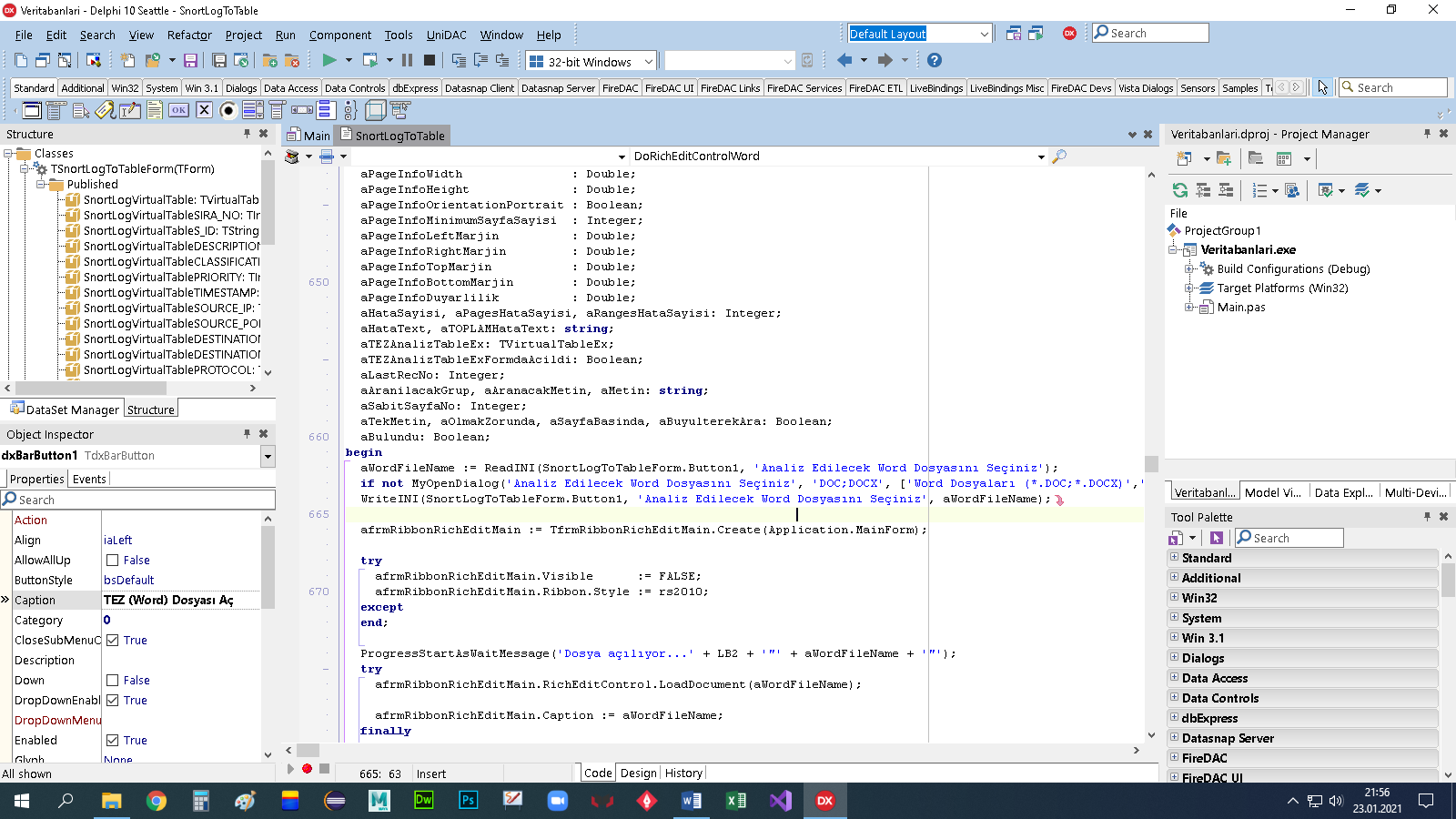
* Projenin hedeflenen 12 haftalık sürede tamamlanması,
* Tüm tez dokümanındaki objeleri veritabanına en kısa sürede kayıt edebilemeli,
* Gereksinim Analizinde tespit edilen hususlara ait tanımlanan tüm kabiliyetleri (sistem genelinde elverişlilik, kullanıcı dostu arayüz) sağlıyor olması

temel başarım gerekleri olarak belirlenmiştir.

3.3 Arayüz (Modül) Gerekleri

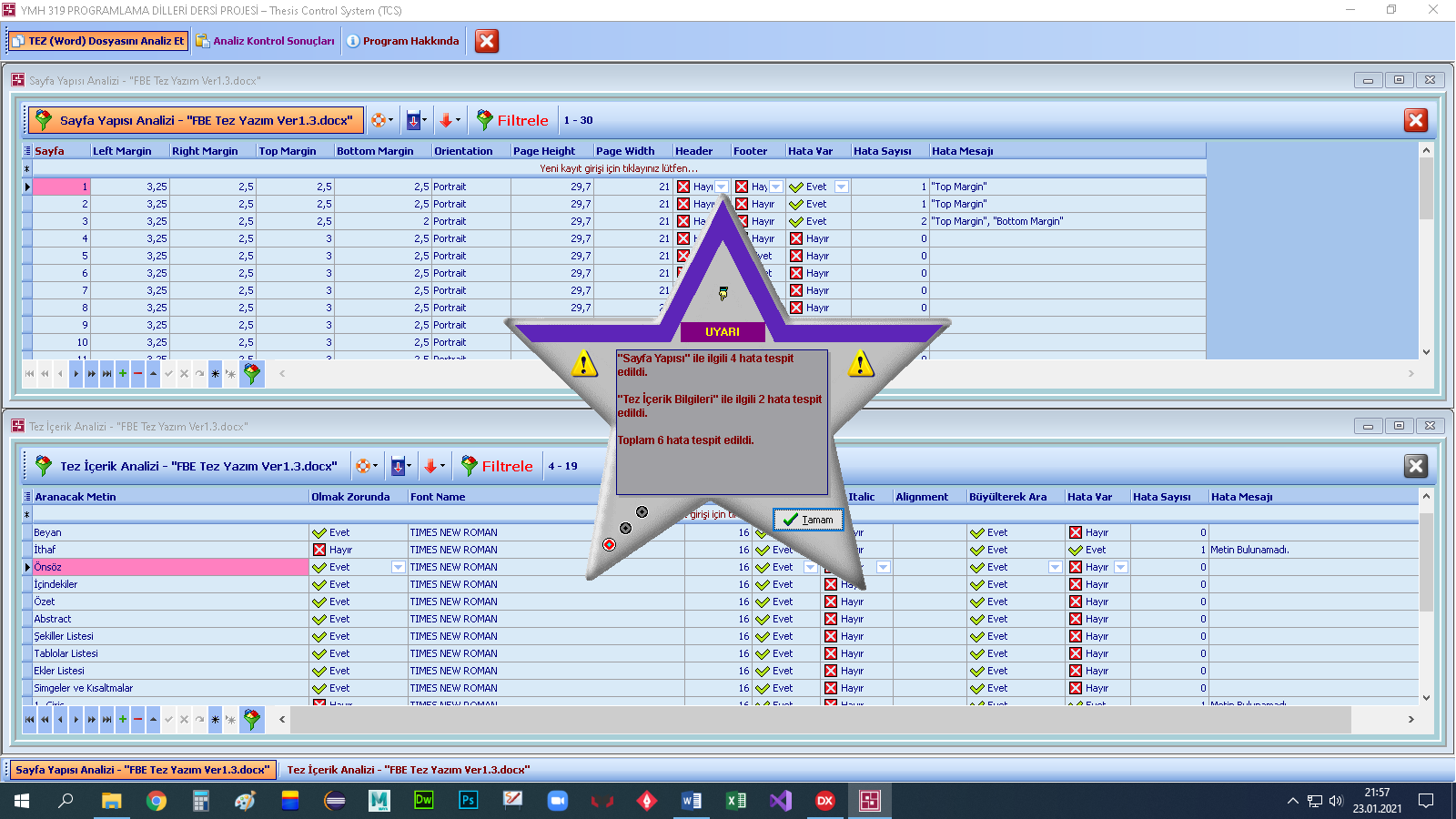
3.3.1 Yazılım Arayüzü





Şekil 11 Main Form

## 3.3.2 Kullanıcı Arayüzü



Şekil 12 Tez Kontrol Sistemi Kullanıcı Arayüzü

# 3.4 Belgeleme Gerekleri

## 3.4.1 Geliştirme Sürecinin Belgelenmesi

Proje Kapsam Planı, İş-Akış şemaları Proje yöneticisi tarafından yapılmıştır.

Sürecin test aşamasında karşılaşılan bir hata ile ilgili, gönderilen Rapor aşağıdadır: Rapor sonucunda yapılan düzenleme ile yazılım stabil halde çalışmasına devam etmektedir:

|  |  |  |
| --- | --- | --- |
| YAZILIM RAPORU | | |
| Yazılım Tanımı |  |  |
| Programcı Ad-Soyad | Abdullah AYGÜN |
| Programlama Dili | Pascal (Delphi) |
| Bilgisayar Sistemi | Intel i7 |
| İşletim Sistemi | Windows10 |
| Kaynak Kod Satırı | 137 |
| Yazılım Tasarımı | Problemin Tanımlanması | Her bir objeyi ayrı bir listeye aktarıp veritabanına kaydetmesinin çok uzun sürmesi. |
| Problemin Çözümü | Belgeyi ilk önce XML formatına çevirip daha sonra işlem adımlarına geçildiğinde sürenin minimuma indiği gözlemlenmiştir. |
| Sonuçların Testi | | Test sonucu ilgili kod dosyaya eklendi ve sorun çözüldü. |

## 3.4.2 Eğitim Belgeleri

Proje kapsamında, kullanıcıların 1 saatlik eğitim ile yeni sisteme adapte olmaları amaçlanır. Genel Windows ve veritabanı kontrolleri eğitimi, proje kapsamında değildir.

## 3.4.3 Kullanıcı El Kitapları

|  |  |
| --- | --- |
|  | Tez Kontrol Sisteminin Kullanımı |
| 1. Açmak istediğiniz tez dokümanını seçiniz ve programın uygun çıktıları vermesini bekleyiniz. |  |
| 1. Analiz Kontrol menüsü ile sonuçları görüntüleyin |  |

4. SİSTEM TASARIMI

# 4.1 Genel Tasarım Bilgileri

## 4.1.1 Genel Sistem Tanımı

Belirli bir tez şablonuna göre tez çalışması yapan kişinin tez yazarken yaptığı hataları minimuma indirgemek amaçlanmaktadır.

Program lokal olarak çalışmakta tez şablonundaki kuralları yerel bilgisayarda gerçek zamanda olmaksızın detaylı bir şekilde istenilen çıktıyı üretebilmektedir.

Gereksinim isteklerinin analizi sonucu; programın sağladığı detaylı filtreleme özelliklerinden dolayı maliyet/etkinlik açısından en mantıklı seçenek olduğu kararına varıldı.

## 4.1.2 Varsayımlar ve Kısıtlamalar

Proje, çok karmaşık bir şekilde yapılan hataları algılayamamaktadır. Bu nedenle hatanın ne kadar büyük olduğu bir yetersizlik olarak görülmektedir.

## 4.1.23 Sistem Mimarisi

|  |  |
| --- | --- |
| C:\Users\AYGUN\AppData\Local\Microsoft\Windows\INetCache\Content.Word\deneme (1).png  Şekil 17 Genel Sistem Mimarisi | Hata çıktısı için ilk önce istenen bir tez dokümanını XML formatına dönüştürülerek gerekli değerler okunmaya başlanır.  Herhangi bir word belgesi ISO/IEC29500 formatında yazıldığından bu işlem her türlü word belge için geçerli olmaktadır.  Üniversitemiz tarafından belirlenmiş olan tez yazım şablonundaki kurallar dikkate alınarak kural yapısı oluşturulup veritabanına aktarılmaktadır.  Aktarılmış olan bu kurallara göre Tez Kontrol Sistemi çalıştırılarak kullanıcıya istenen çıktıları vermektedir. |

## 4.1.4 Dış Arabirimler

### 4.1.4.1 Kullanıcı Arabirimleri

Program herhangi bir dış arabirime ihtiyaç duymamaktadır.

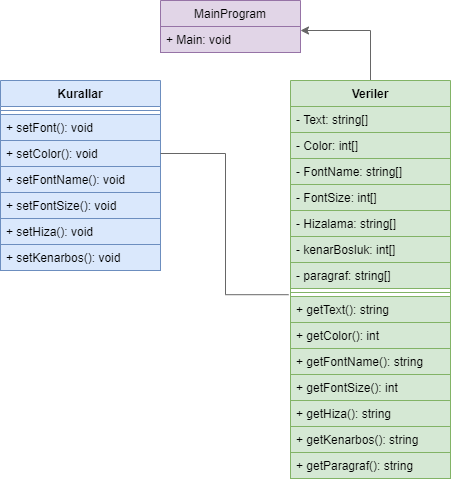
### 4.1.4.2 Veri Arabirimleri

SQLLite, veritabanımız biçimsel yazım kuralları ile tez metninde Nesne niteliğindeki özellikleri tutmak için kullanılacaktır.

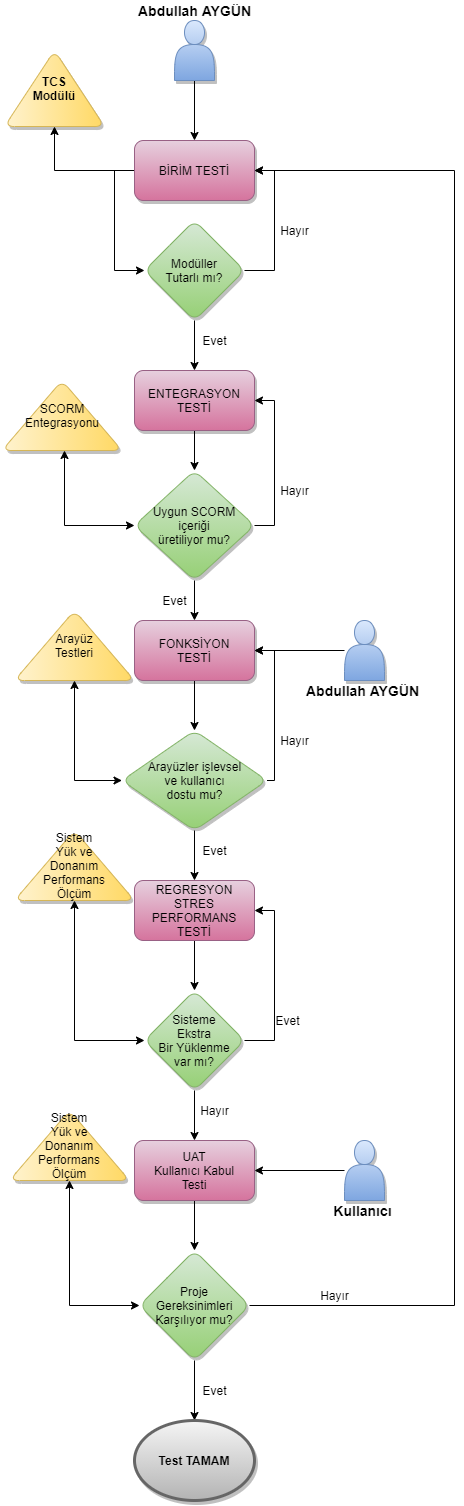
### 4.1.4.3 Diğer Sistemlerle Arabirimler

Projemizde yukarıda sayılanlar dışında başka bir dış arabirim kullanılmamıştır.

## 4.1.5 Veri Modeli

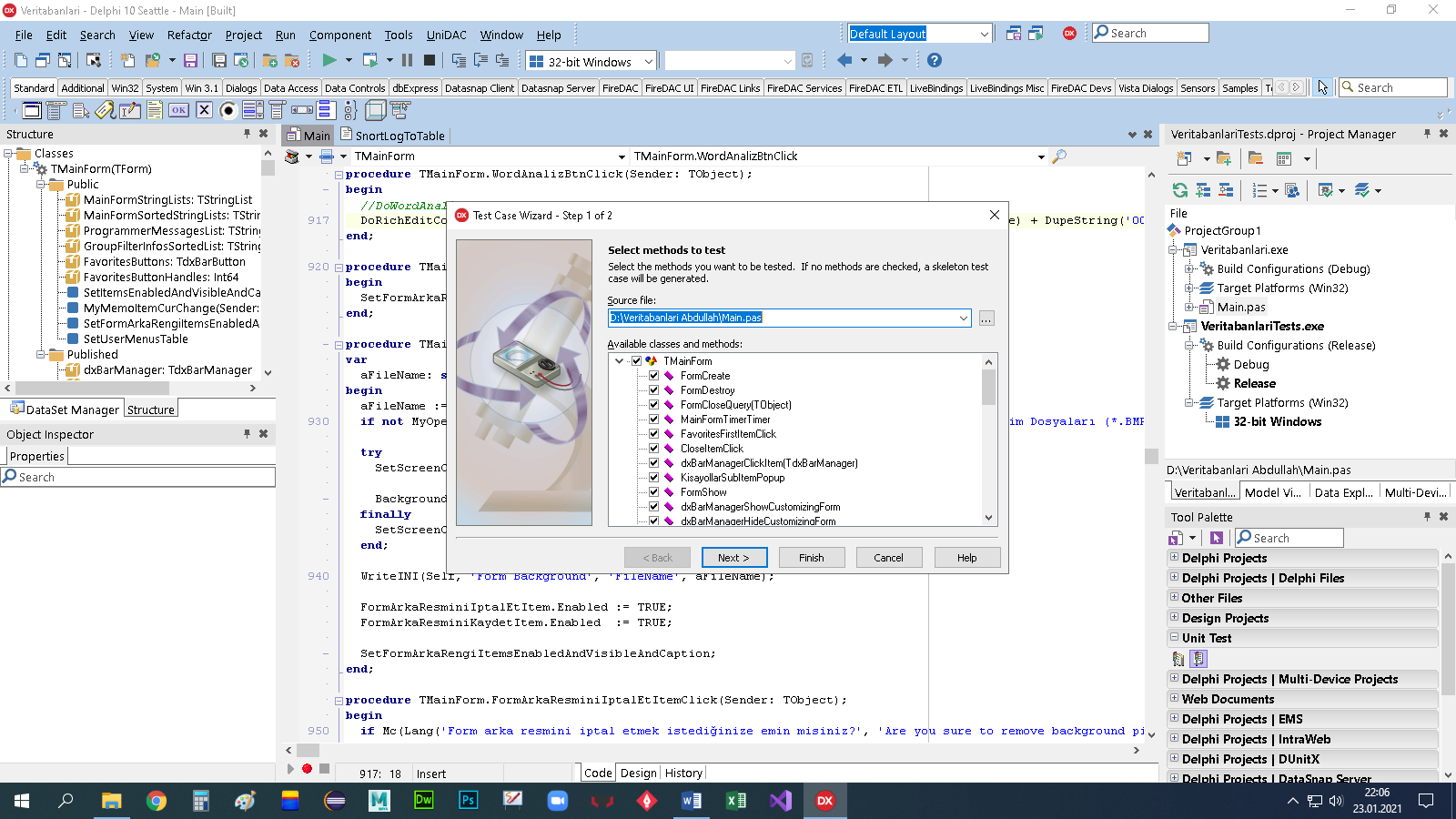


## 4.1.6 Testler

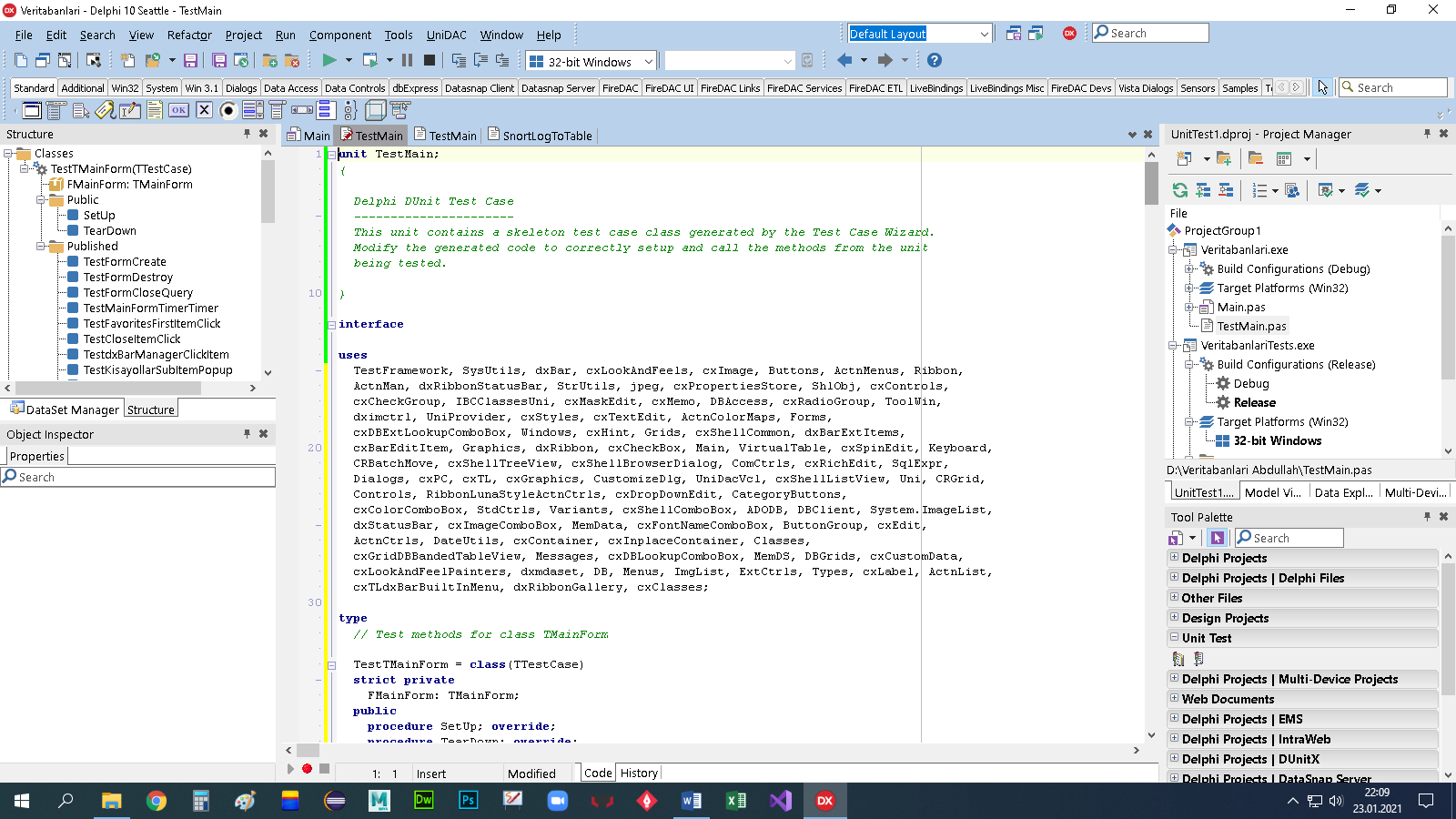


## 4.1.7 Performans

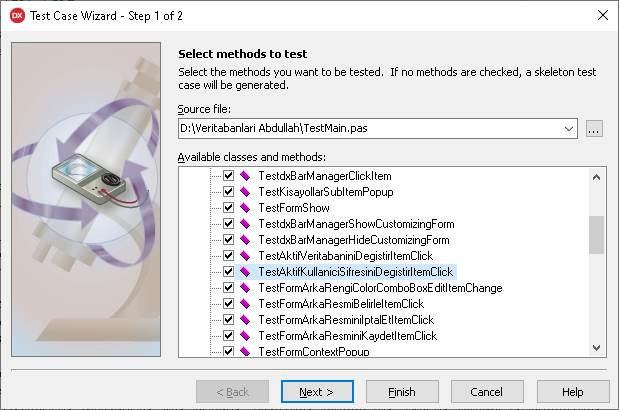
Performans test planında; özellikle stres testi için Delphi Unit Test aracı kullanıldı. Test için yapılan iş tanımında “İş parçacığı sayısı” ve “Rampa süresi” = 10000, “Döngü sayısı” = 500 alındı.



Şekil 19 Ubit Test Thread Grup Ayarlaması



Şekil 20 Test Case Oluşturulması

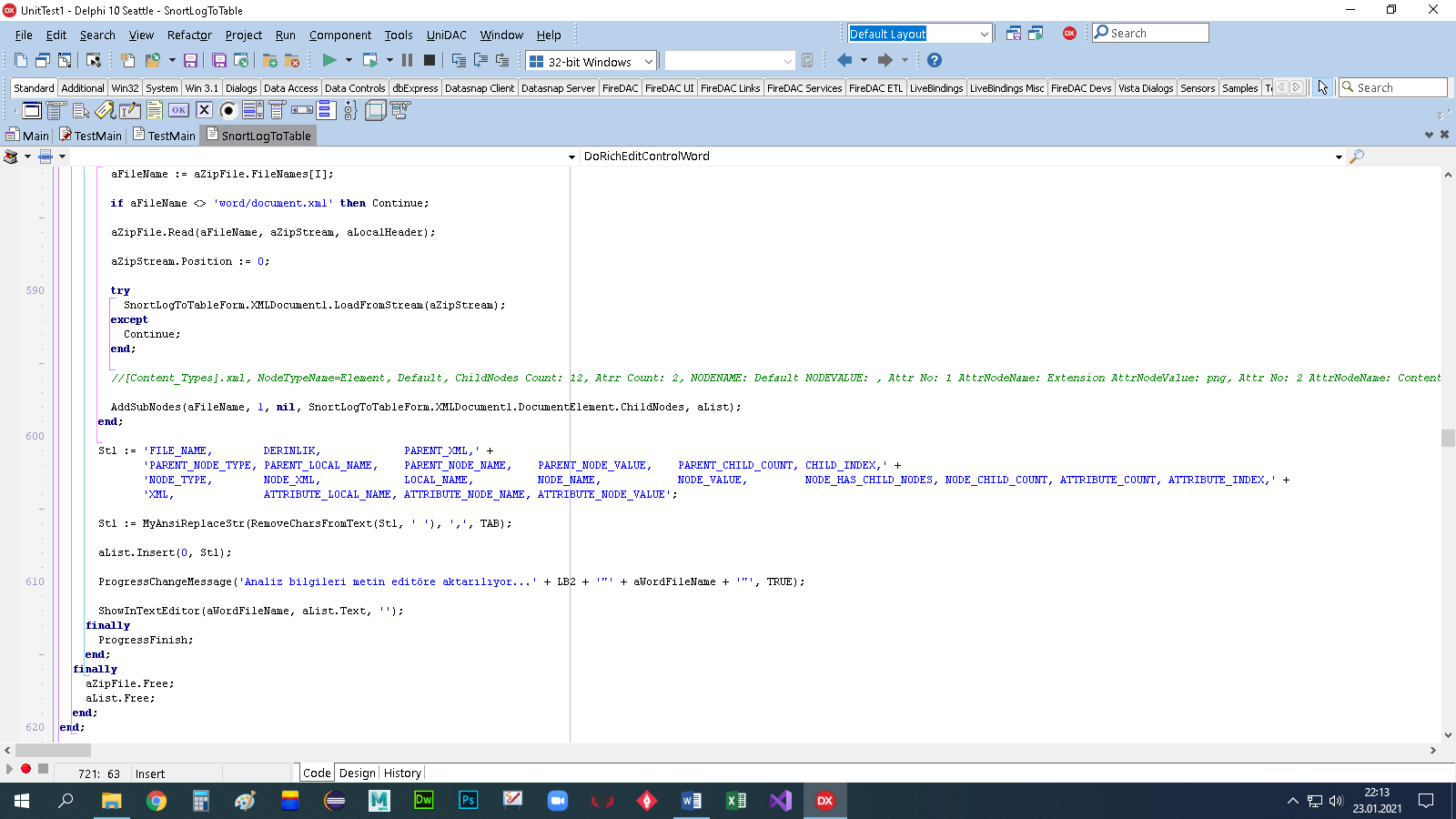


Şekil 21 iş parçactğt Test Event

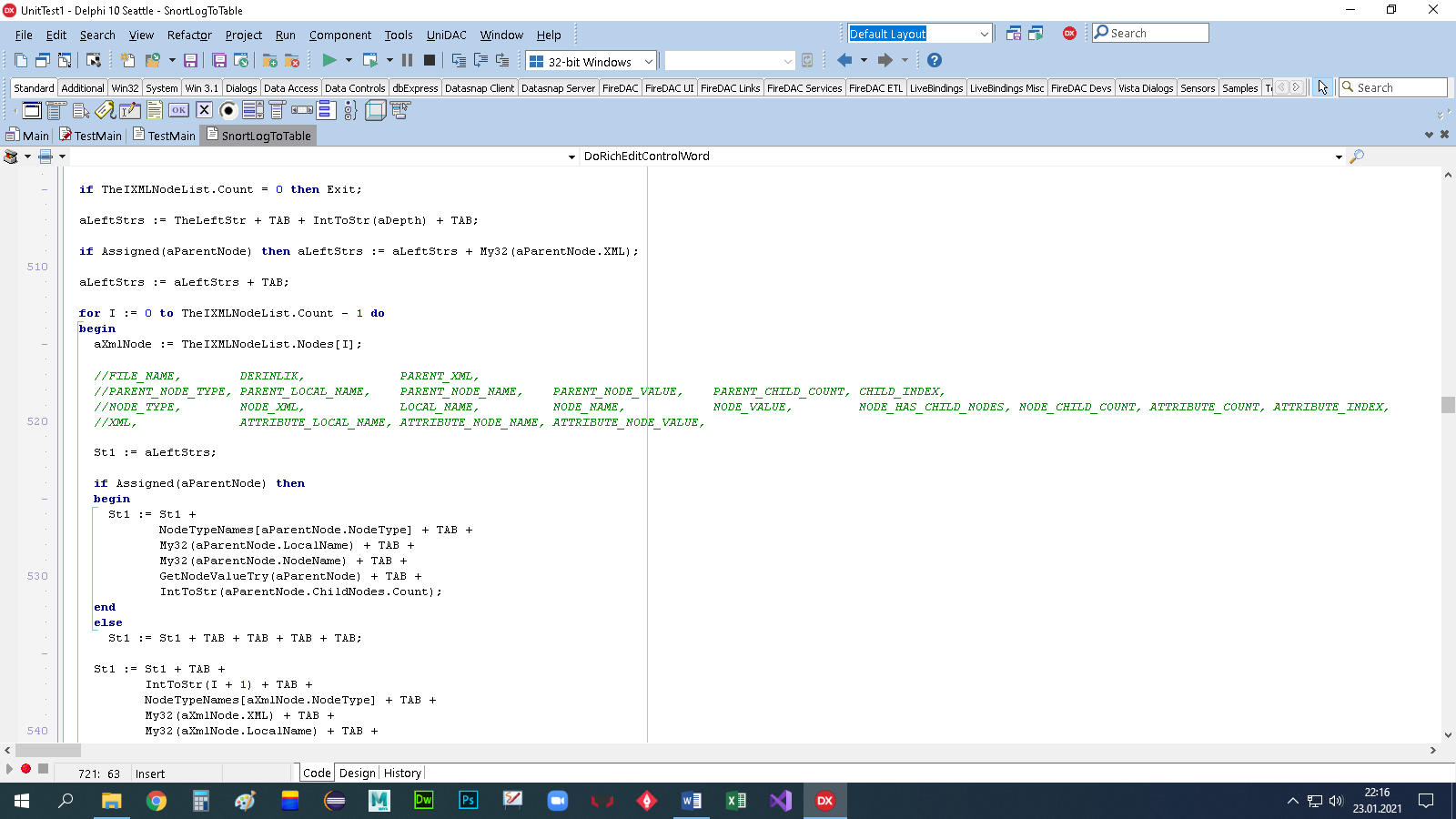
# 4.2 Veri Tasarımı

OpenXML ile okunan docx verileri custom bir sanal tabloya aktarılmaktadır. Tablo yapısı koda gömülü hakdedir.

## 4.2.1 Tablo tanımları



Şekil 23 Tablo yapısı



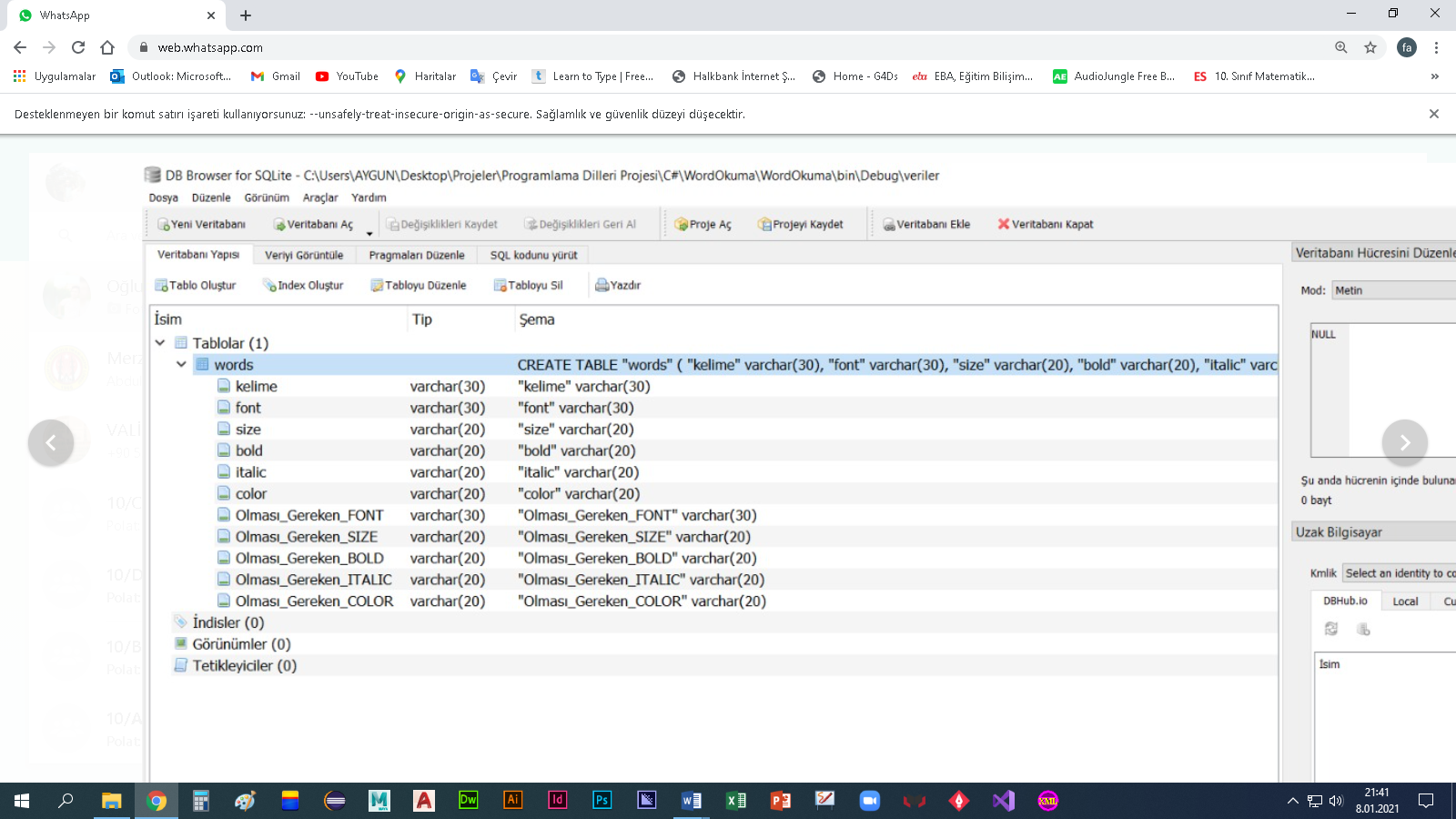
## 4.2.2 Tablo- İlişki Şemaları

Program tek tablodan oluştuğu için ilişkisel bir yapı bulunfutmamaktadır.

## 4.2.3 Veri Tanımları

## 

## 4.2.4 Değer Kümesi Tanımları



# 4.3 Süreç Tasarımı

## 4.3.1 Genel Tasarım

Gereksinim Analizi yapıldıktan sonra, OpenXML kütüphanesinin kullanımının zaman ve etkinlik açısından uygun olacağı değerlendirildi.

Veritabanı yapısında tek tablo olacağı düşünülerek Virtual bir tablo kullanımına karar verildi.

Yazım kurallarının analizi ve sınıflandırılması yapılarak kodlanmaya hazır hale getirildi.

## 4.3.2 Modüller

### 4.3.2.1 Giriş Modülü (Login)

Program tek kullanıcılı ve lokal çalışan bir uygulama olduğundan giriş modülüne ihtiyaç duyulmamıştır.

### 4.3.2.2 Kullanıcı Modülü (Ana Menü)

#### 4.3.2.2.1 İşlev

* Kontrolü yapılacak belge yüklenir.
* Yüklenen belgenin uygunluğu program tarafından kontrol edilecek, işlem sonuçlandırılır.

## 4.3.4 Entegrasyon ve Test Gereksinimleri

Proje kapsamında, gereksinimleri karşılamak üzere başka bir entegrasyona ihtiyaç duyulmamıştır.

5. SİSTEM GERÇEKLEŞTİRİMİ

# 5.1. Giriş

Proje, standart Pascal kodlaması ile oluşturulan tek formlu yapıdan oluşmaktadır. Veritabanı olarak koda gömülü virtual tablo kullanılmış, kod yazımı için Delphi 10 Seattle IDE kullanılmıştır.

# 5.2. Yazılım Geliştirme Ortamları

## 5.2.1 Programlama Dilleri

Standart Pascal WordProcessingML namespace, ve Open XML kütüphanesi kullanılarak ana çatı oluşturulmuş, sanal veritabanı ile kurallar ve doküman içerikleri tutulmuştur.

## 5.2.2 Veri Tabanı Yönetim Sistemleri

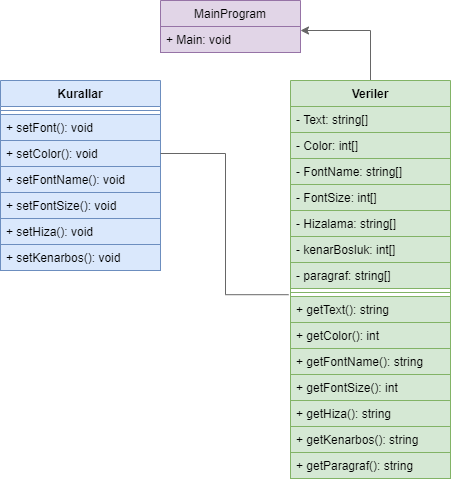
### 5.2.2.1 VTYS Kullanımının Ek Yararları

* Esneklik
* Bilgileri kolaylıkla güncelleyebilme
* Ortak verilerin tekrarının önlenmesi
* Yazılım geliştirmenin ve veri depolamanın kolaylaşması
* Geliştirme zamanının azalması
* Güncel bilgilerin tüm kullanıcılara aynı anda ulaşması

### 5.2.2.2 Veri Modelleri

Proje, standart Pascal kodlaması ile oluşturulan tek formlu ve tek tablolu bir yapıdan oluşmaktadır. Veritabanı olarak sanal tablo kullanılmıştır.

### Şemalar



Şekil 37Veritabanı Şemalar

### 5.2.2.4 Hazır Program Kütüphane Dosyaları

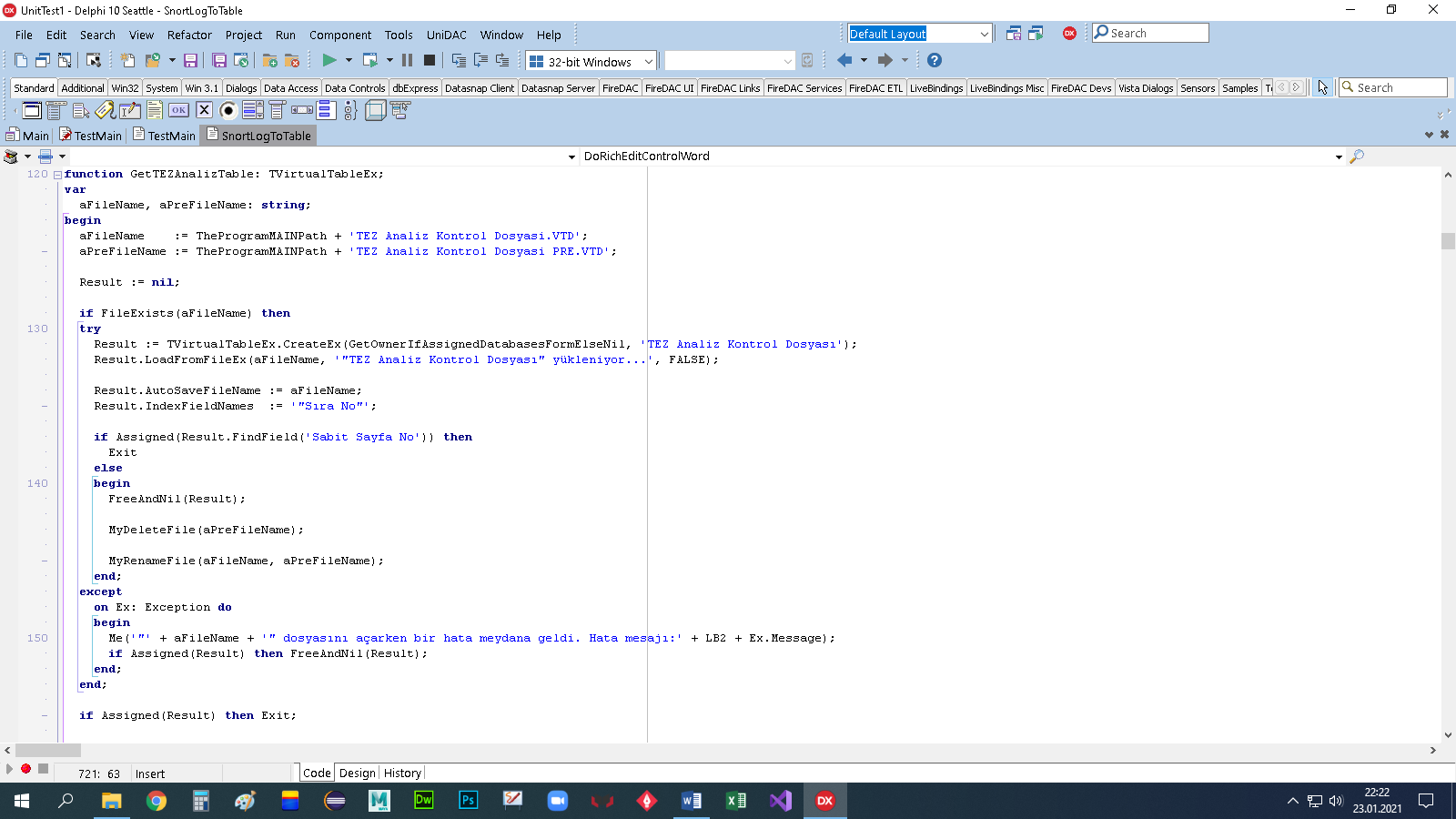
OpenXML kütüphanesi ve Pascal, DevExpress, native WordProcessingML modülü kullanılmıştır

### 5.2.2.4 CASE Araç ve Ortamları

CASE diyagramları için <https://app.diagrams.net/> sitesindeki online araçlar ve MS Visio kullanılmıştır.

# 5.3. Kodlama Stili

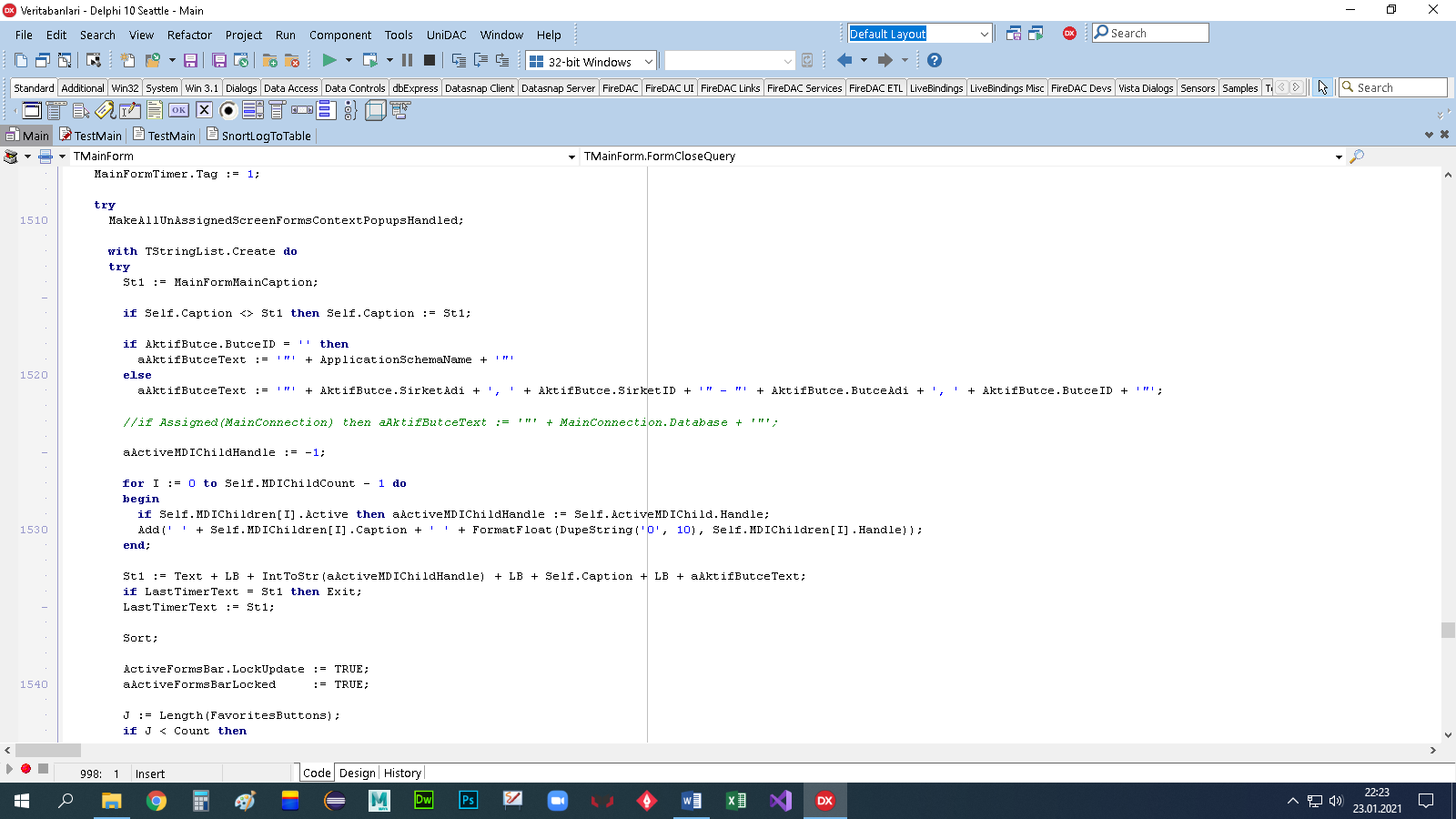
Kodlamamızı, Delphi 10 Seattle ile yaptık. IDE’nin varsayılan Pascal kod stilini kullandık, herhangi bir özelleştirme yapmadık.



Şekil 39Proje kodlamasıIDE Kod Stili

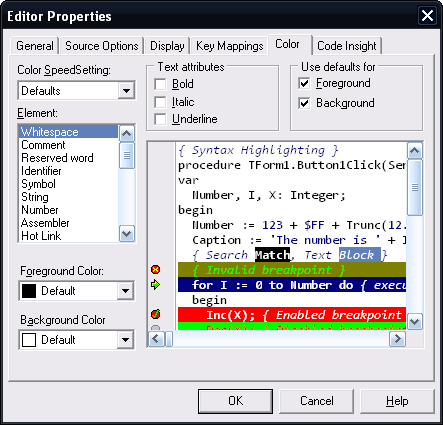
## 5.3.1 Açıklama Satırları

Açıklama Satırları Visual Studio IDE’nin varsayılan template ile yapıldı, herhangi bir özelleştirme yapılmadı.



Şekil 40 Açıklama Satırlarının Kullanımı

## 5.3.2 Kod Biçimlemesi



Şekil 41 Delphi 10 Seattle IDE Coding Şablonu

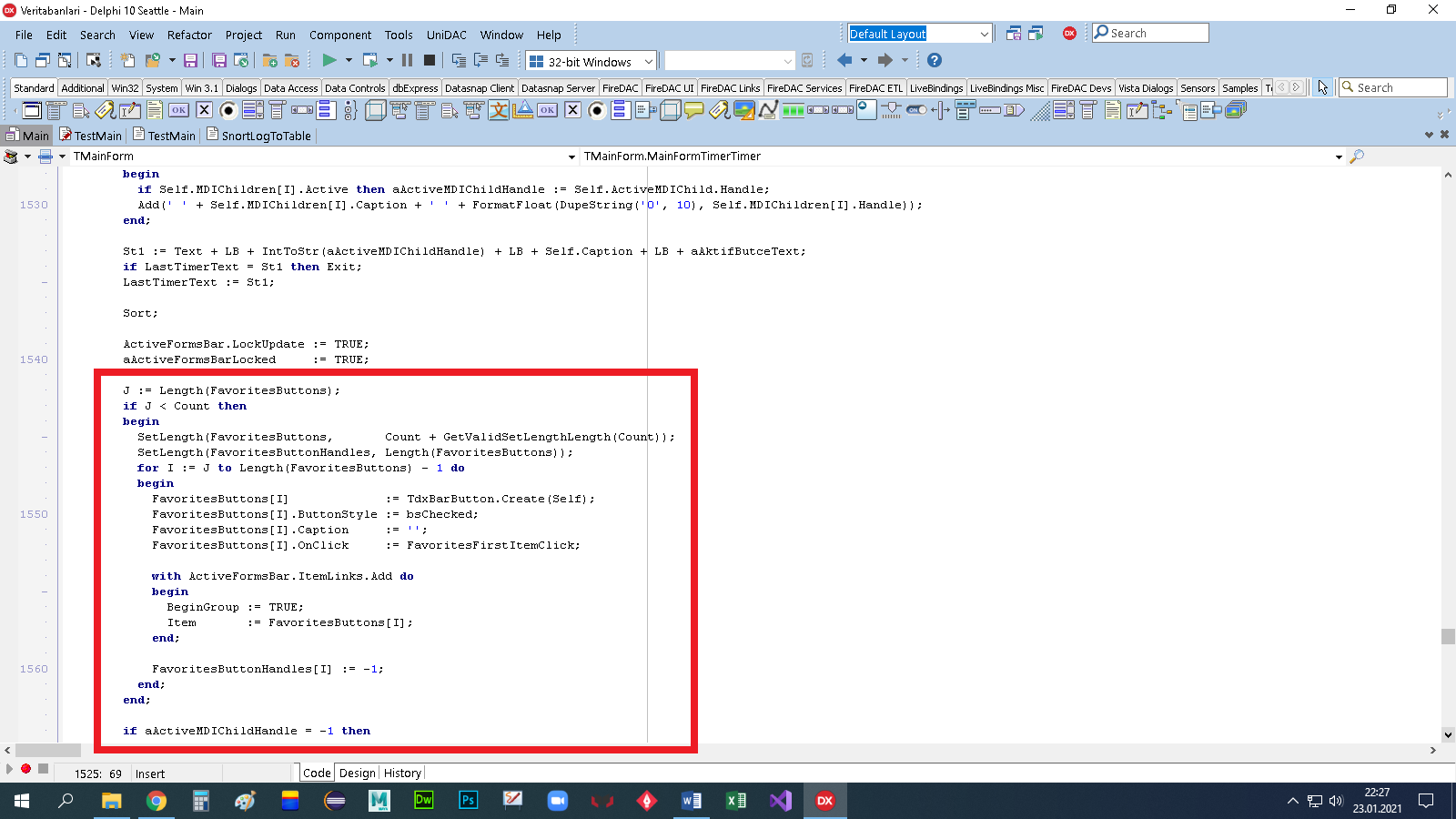
## 5.3.3 Anlamlı İsimlendirme

Pascal Standard Tavsiye Kuralları (PSR-1) çerçevesinde; sınıf adları Pascal Case (NesneIsmi); sınıf sabitleri “MAKRO\_SABIT”; metot tanımları camel Case (getString) şeklindedir.

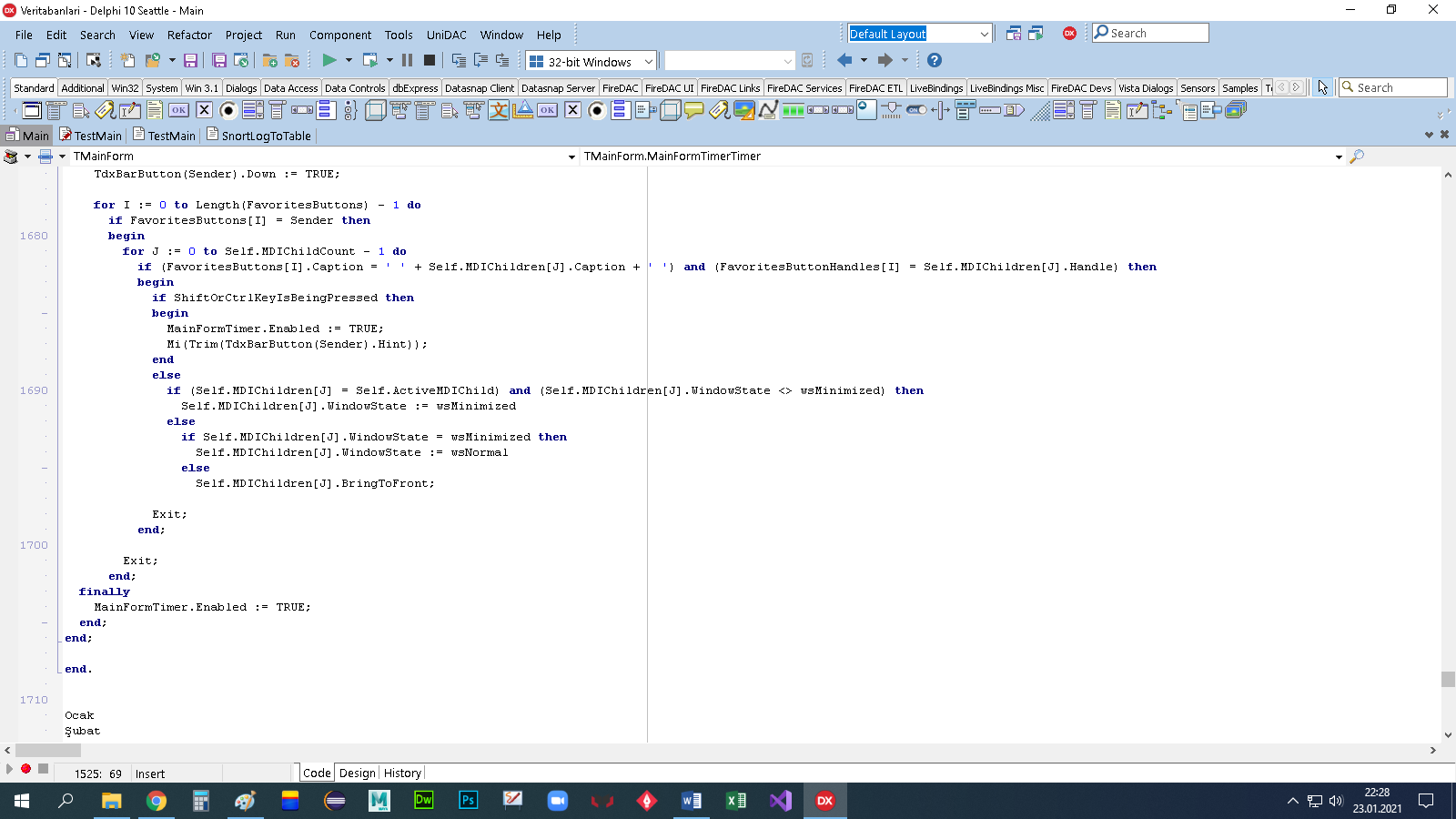
## 5.3.4 Yapısal Programlama Yapıları

Ardışık işlem yapıları: Bu tür yapılarla genellikle fonksiyon, altprogram ve buna benzer tekrarlı yapıları tek bir seferde çözdük.

Koşullu işlem yapıları: Bu yapıları ise neredeyse programın tamamında kullandık karşılaştırma yapılan her yerde bunlara yer verildi.



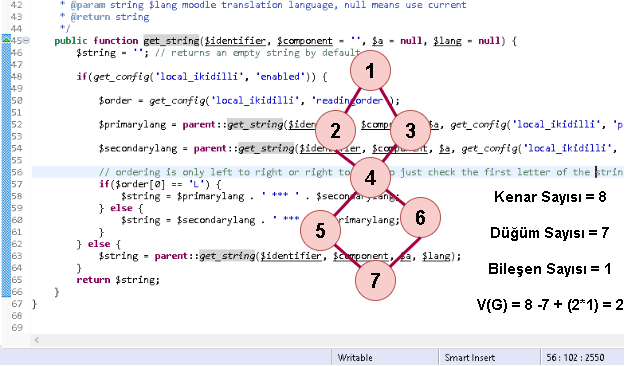
Döngü yapıları: Tıpkı ardışık işlemler gibi alt alta birkaç satır yazıcığımıza tek bir döngüyle bu sorunların üstesinden geldik.



# 5.4. Program Karmaşıklığı

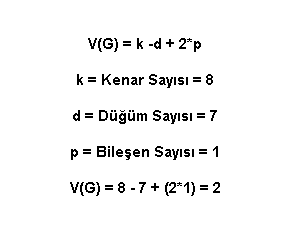
McCabe karmaşıklık ölçütü kullanılmıştır. McCabe karmaşıklık ölçütünün hesaplanmasından önce programın çizge biçimine dönüştürülmesi gerekmektedir.

## 5.4.1 Programın Çizge Biçimine Dönüştürülmesi



Şekil 42 McCabe Metodu ile Programa karmaşıklığının bulunması

## 5.4.1 McCabe Karmaşıklık Ölçütü Hesaplama



# 5.5. Olağan Dışı Durum Çözümleme

Olağan dışı durum, bir programın çalışmasının, geçersiz ya da yanlış veri oluşumu ya da başka nedenlerle istenmeyen bir biçimde sonlanmasına neden olan durum olarak tanımlanmaktadır.

## 5.5.1 Olağandışı Durum Tanımları

Moodle Site için, olağan dışı durumları bir unsurun;

* Çalışmaması
* Yavaş çalışması
* Beklendiği gibi çalışmaması

Noktalarından değerlendirebiliriz.

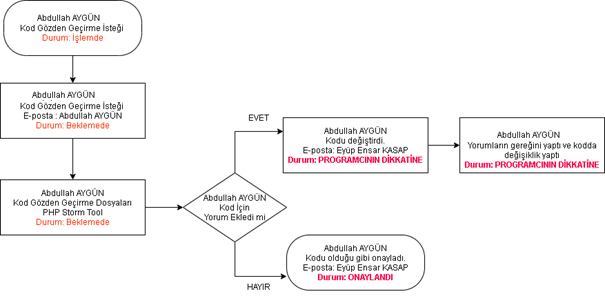
|  |  |  |  |
| --- | --- | --- | --- |
| Unsur Tanımı | Çalışmıyor | Yavaş Çalışıyor | Beklendiği gibi çalışmıyor |
| Ana Menü | Kod kontrolü yapılacak | Kod Optimizasyonu | Semantik kontrolü |

## 5.5.2 Farklı Olağandışı Durum Çözümleme Yaklaşımları

Özgün Tez Yazım belgesinin biçimsel yapısından çok uzak olan belgelerin kontrolü performans düşüklüğüne ve beklenmeyen çıktılar elde edilmesine sebep olabilir. Bu sebeple, kontrol edilecek belgenin ana hatlarıyla özgün şablon belgeye uyması beklenir.

# 5.6. Kod Gözden Geçirme

## 5.6.1 Gözden Geçirme Sürecinin Düzenlenmesi



## 5.6.2 Gözden Geçirme Sırasında Kullanılacak Sorular

### 5.6.2.1 Öbek Arayüzü

|  |  |
| --- | --- |
| Gözden Geçirme Konusu | Ana Form |
| Öbek tek bir işlevsel amacı yerine getiriyor mu? | ✓ |
| Öbek adı, işlevini açıklayacak biçimde anlamlı olarak verilmiş mi? | ✓ |
| Öbek tek giriş ve tek çıkışlı mı? | ✓ |
| Öbek bir işlev ise, parametrelerinin değerini değiştiriyor mu? | ✓ |

### 5.6.2.2 Giriş Açıklamaları

|  |  |
| --- | --- |
| Gözden Geçirme Konusu | Ana Form |
| Öbek, doğru biçimde giriş açıklama satırları içeriyor mu? | ✓ |
| Giriş açıklama satırları, öbeğin amacını açıklıyor mu? | ✓ |
| Giriş açıklama satırları, parametreleri, küresel değişkenleri içeren girdileri ve kütükleri tanıtıyor mu? | - |
| Giriş açıklama satırları, öbekte yapılan değişikliklere ilişkin tanımlamaları içeriyor mu? | - |
| Giriş açıklama satırları, öbekteki olağan dışı durumları tanımlıyor mu? | - |
| Her paragrafı açıklayan kısa açıklamalar var mı? | ✓ |

### 5.6.2.3 Veri Kullanımı

|  |  |
| --- | --- |
| Gözden Geçirme Konusu | Ana Form |
| İşlevsel olarak ilintili bulunan veri elemanları uygun bir mantıksal veri yapısı içinde gruplanmış mı? | ✓ |
| Değişken adları, işlevlerini yansıtacak biçimde anlamlı mı? | ✓ |
| Değişkenlerin kullanımları arasındaki uzaklık anlamlı mı? | ✓ |
| Her değişken tek bir amaçla mı kullanılıyor? | ✓ |
| Dizin değişkenleri kullanıldıkları dizinin sınırları içerisinde mi tanımlanmış? | ✓ |
| Tanımlanan her gösterge değişkeni için bellek ataması yapılmış mı? | ✓ |

### 5.6.2.4 Öbeğin Düzenlenişi

|  |  |
| --- | --- |
| Gözden Geçirme Konusu | Ana Form |
| Modüller birleşimi uyumlu mu? | ✓ |
| Modüller arası veri aktarımları sağlanıyor mu? | - |
| Bütün modüller birleştiğinde sistem çalışıyor mu? | ✓ |

### 5.6.2.5 Sunuş

|  |  |
| --- | --- |
| Gözden Geçirme Konusu | Ana Form |
| Her satır, en çok bir deyim içeriyor mu | ✓ |
| Bir deyimin birden fazla satıra taşması durumunda, bölünme anlaşılabilirliği kolaylaştıracak biçimde anlamlı mı? | ✓ |
| Koşullu deyimlerde kullanılan mantıksal işlemler yalın mı? | ✓ |
| Bütün deyimlerde, karmaşıklığı azaltacak şekilde parantezler kullanılmış mı? | ✓ |
| Bütün deyimler, belirlenen program stiline uygun olarak yazılmış mı? | ✓ |
| Öbek yapısı içerisinde akıllı "programlama hileleri" kullanılmış mı? | - |

6. DOĞRULAMA VE GEÇERLEME (TEST)

# 6.1. Giriş

Geliştirilecek bilgi sistemi yazılımının doğrulanması ve geçerlenmesi, üretim süreci boyunca süren etkinliklerden oluşur. Söz konusu etkinlikler:

* Yazılım belirtimlerinin ve proje yaşam sürecindeki her bir etkinlik sonunda alınan çıktıların, tamam, doğru, açık ve önceki belirtimleri tutarlı olarak betimler durumda olduğunun doğrulanması.
* Proje süresince her bir etkinlik ürününün teknik yeterliliğinin değerlendirilmesi ve uygun çözüm elde edilene kadar aktivitenin tekrarına sebep olması. Projenin bir aşaması süresince geliştirilen anahtar belirtimlerin önceki belirtimlerle karşılaştırılması.
* Yazılım ürünlerinin tüm uygulanabilir gerekleri sağladığının gerçeklenmesi için sınamaların hazırlanıp yürütülmesi biçiminde özetlenebilir.



Şekil 43Doğrulama ve Geçerleme

# 6.2 Sınama Kavramları

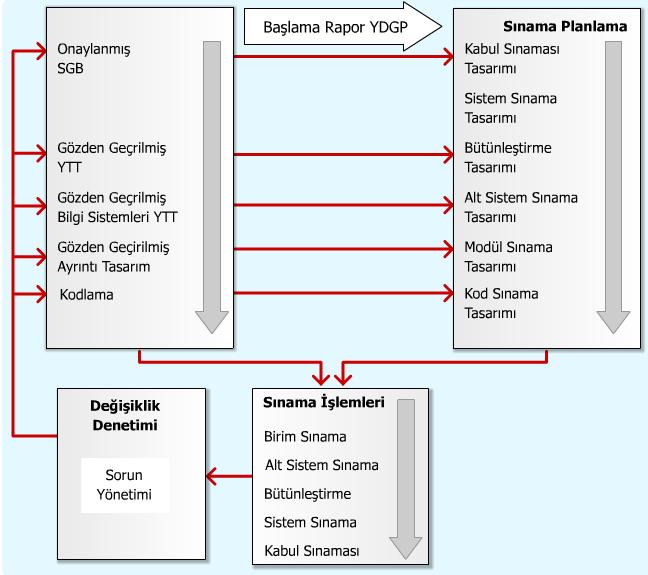
**Birim Sınama:** Projenin mevcut Ana Formunun sınanmasıdır.

**Alt Sistem Sınama:** Alt Sistem kullanılmadığından sınama süreci olmayacaktır.

**Sistem Sınama:** Hata ya da performans sorunu ortaya çıkmadı.

**Kabul Sınama:** Proje gereksinimlerini karşıladığı, performansta bir düşüş olmadığı görüldü.

# 6.3. Doğrulama ve Geçerleme Yaşam Döngüsü

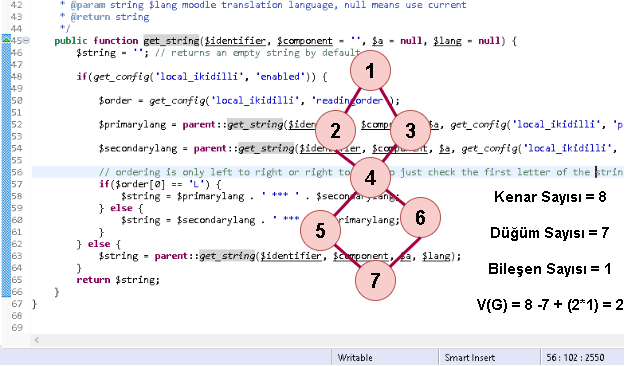


# 6.4. Sınama Yöntemleri

## 6.4.1 Beyaz Kutu Sınaması

|  |  |  |
| --- | --- | --- |
| Sınama Noktası | Durum | Sınamayı Yapan |
| Bütün bağımsız yolların en azından bir kere sınanması, | ✓ | Abdullah AYGÜN |
| İç veri yapılarının denenmesi | ✓ |
| Bütün döngülerin sınır değerlerinde sınanması | ✓ |
| Bütün mantıksal karar noktalarında iki değişik karar için sınamaların yapılması, | ✓ |

## 6.4.2 Temel Yollar Sınaması



Toplam Dal Sayısı (E) = N (Toplam düğüm sayısı) + 2 🡺 E = 7 +2 = 9

# 6.5. Sınama ve Bütünleştirme Stratejileri

## 6.5.1 Yukarıdan Aşağı Sınama ve Bütünleştirme

## 6.5.2 Aşağıdan Yukarıya Sınama ve Bütünleştirme

# 6.6. Sınama Planlaması

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| ZAMAN  İŞ | Proje Başlangıç Tarihi:01/02/2021 | | |  | |  | | | | | | | |
| **10. Hafta** | | | | | | | | **11.Hafta** | | | | | | | | | |
| Sistem Doğrulanması | **Birim Sınama** | | | | | | | **Alt Sistem Sınama** | | | | **Sistem Sınama** | | | | **Kabul Sınama** | | |
| 1 | 2 | 3 | | 4 | | 5 | 6 | | 7 | 1 | 2 | 3 | | 4 | 5 | 6 | 7 |
|  |  |  | |  | |  |  | |  |  |  |  | |  |  |  |  |
| Ana Form sınama | | | | | | | Planlanmadı | | | | Sistem yük, stres, performans testi uygulandı. | | | | UAT (Kullanıcı Kabul Testi) yapıldı. Sistem protipten çıkarıldı. | | |

# 6.7. Sınama Belirtimleri

|  |  |  |
| --- | --- | --- |
| YAZILIM SINAMA RAPORU | | |
| Yazılım Tanımı |  |  |
| Programcı Ad-Soyad | Abdullah AYGÜN |
| Programlama Dili | Pascal |
| Bilgisayar Sistemi | Intel i7 |
| İşletim Sistemi | Windows10 |
| Sınama Türü | Beyaz Kutu |
| Yazılım Tasarımı | Problemin Tanımlanması | Her bir objeyi ayrı bir listeye aktarıp veritabanına kaydetmesinin çok uzun sürmesi. |
| Problemin Çözümü | Belgeyi ilk önce XML formatına çevirip daha sonra işlem adımlarına geçildiğinde sürenin minimuma indiği gözlemlenmiştir. |
| Sonuçların Testi | | Test sonucu ilgili kod sunucudaki dosyaya eklendi ve sorun çözüldü. |

# 6.8. Yaşam Döngüsü Boyunca Sınama Etkinlikleri

7. BAKIM

# 7.1 Giriş

* Projemize ait modüllerin “Düzeltici Bakımı” Doğrulama ve Geçerleme sürecinde gerçekleştirilen testlerle yapıldı. Diğer tür bakımlar ise; progtamın versiyon yükseltilmesi ile gerçekleşmiş olacak. Bu noktada bakım ile ilgili kodlama yapmamızı gerekytiren ekstra bir işlem bulunmamaktadır.

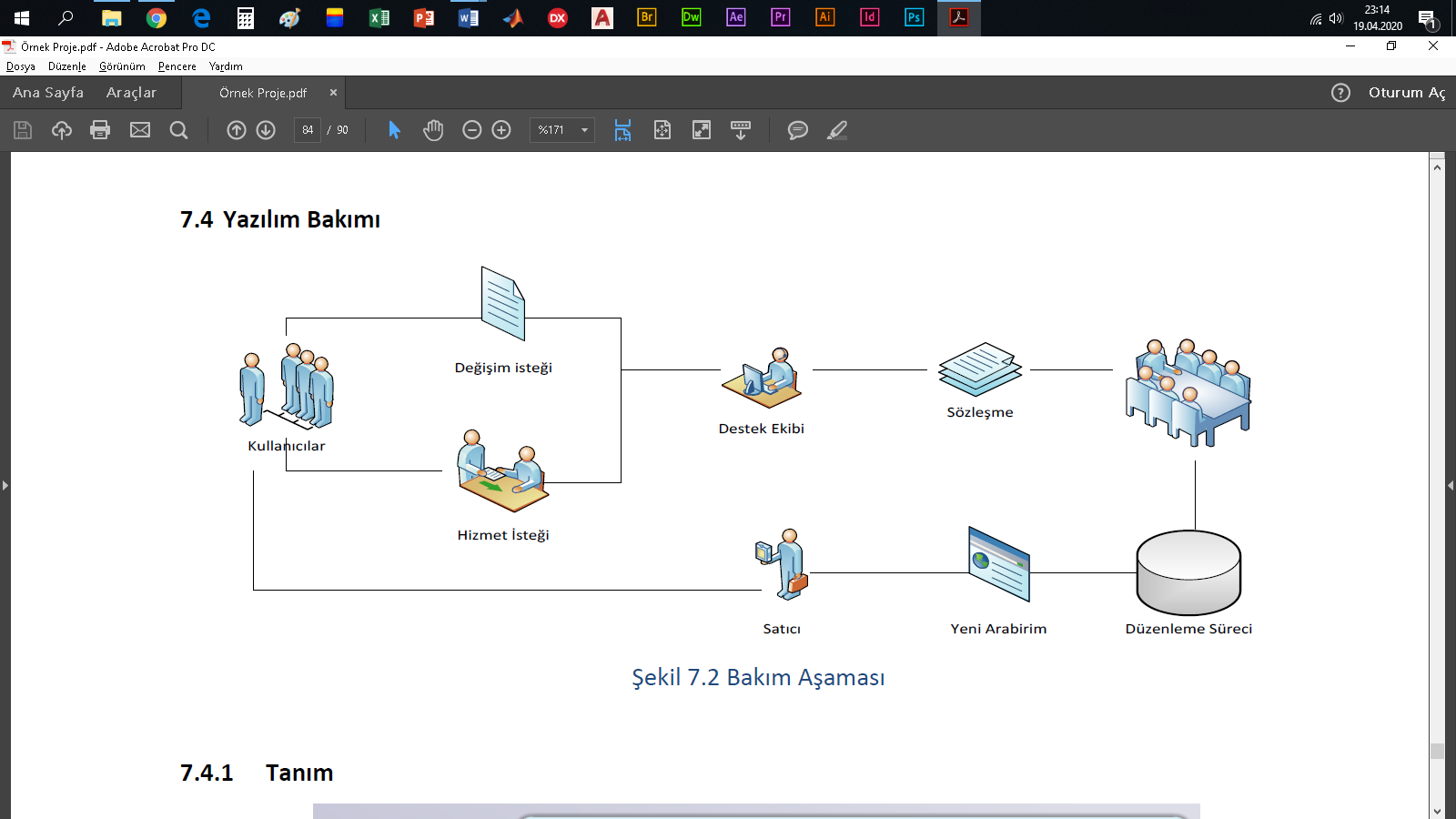
# 7.2 Kurulum

Proje, kurulum gerektirmemektedir. Exe dosyanın uygun konuma yapıştırılması ile program çalışmaya başlar.

# 7.3 Yerinde Destek Organizasyonu

Özel bir destek hizmetine ihtiyaç duyulmakaktadır.

# 7.4 Yazılım Bakımı



## 7.4.1 Tanım

Bir yazılımın bakımı(software maintenance), yazılımın geliştirilmesi tamamlandıktan sonraki süreçte, yazılımın ömrünü sağlıklı bir şekilde sürdürmesini sağlayan yaklaşımlar bütünüdür. Yazılım bakımı Yazılım Geliştirme Sürecinin(Software Development Process) son parçasını oluşturur. Müşteri gereksinimlerine göre oluşturulan yazılım çıktısı, değişen teknolojik araçlara ve müşteri gereksinimlerine göre sürekli bakımının yapılması, denetlenmesi gerekir. Nasıl bir arabanın yıllık bakımı yapılıyorsa, yazılım ürünün de belirli sürümler dahilinde geliştirilmesi, iyileştirilmesi ve korunması yapılmalıdır.

## 7.4.2 Bakım Süreç Modeli

IEEE 1219-1998 Yazılım Bakım Standardı esaslarına göre sağlanacaktır.

8. SONUÇ

TCS, proje sonunda, gereksinimlerin karşılayarak proje yaşam döngüsünün tüm süreçlerinden başarı ile çıkarak hizmet vermeye başlamıştır.

Kurulan yeni yazılımın sorunsuz olarak çalıştığı ve ihtiyaç analizinde belirlenen pedagojik ihtiyaçları karşıladıkları belirlendi.

9. KAYNAKLAR

<http://fbe.firat.edu.tr/sites/fbe.firat.edu.tr/files/Tez_Yazim_Kilavuzu_Eylul_2019_Revize_17-07-2020.pdf>

<https://docs.microsoft.com/en-us/office/open-xml/word-processing>

<https://docs.microsoft.com/en-us/dotnet/api/microsoft.office.interop.word?view=word-pia>