Beykent Üniversitesi
Yazılım Mühendisliği Bölümü
Yazılım Mühendisliği Tasarım Projesi

Rapor 2
2023 – GÜZ

Proje İsmi (kısaltması ile birlikte)

Grup Elemanları
No, Adı Soyadı (Alfabetik Sırada)

III Tasarım Dokümantasyonu (16 font)[1]

8 Tasarım Hedeflerinin Tanımlanması (14 font)

Tasarım hedefleri, geliştirilecek sistemin genel tasarımında oldukça etkili olup birtakım ayırt edici özellikler de içerebilir. Örneğin bilgisayar oyunlarında öncelik, çözümün doğruluğundan (accuracy property) ziyade sistemin çalışma hızının yeterince yüksek olmasıdır. Bu nedenle bir bilgisayar oyununda kullanılan motorla, oyunun daha hızlı çalışmasını sağlayacak alternatifler aranabilir; böylece geliştirilen sistemin doğruluğu tasarım hedefi olmayacaktır. Farklı bir yazılım ürününde, örneğin hassas bilimsel hesaplamaların yapıldığı bir çalışmada birinci derecede tasarım hedefi, hızın düşmesi olasılığına rağmen sonuçların doğru elde edilmesi olacaktır.

Tasarım hedefleri ile gereksinimler arasındaki *önemli bir fark vardır*: Gereksinimler, ürünün müşteriye kabul edilebilir olması için gerçekleştirilmesi gereken her şeydir; oysa tasarım hedefleri, tasarımcıların mümkün olan en iyisini yapmak için gerçekleştirmeyi arzuladığı özelliklerdir. Ayrıca, tasarımcılar hedeflerini belirlerken belirli kabul edilebilirlik şartlarını sağlamak zorunda değildir. Oysa gereksinimler müşterinin istekleridir ve yerine getirilmelidir.

Bu bölümde tanımlayacağınız herhangi bir kavramın hem gereksinimler aşamasında, hem de tasarım hedefinde verilmiş olması mümkündür. Bu nedenle bir tasarım hedefi, bir gereksinimin gerçekleştirilmesi ile sistemin mümkün olabildiği kadar hızlı çalışmasını sağlamak olabilir. Ayrıca tasarım hedefine göre sistemin belirli bir eşiğin altındaki bir hızı kabul etmeyeceği ifade edilebilir.

Yazılım ürünü geliştirirken tasarım hedeflerine karar vermede bazı zıt betimlemelerle karşılaşılabilir. Örneğin, bir uygulamanın fonksiyonelliği (functionality), kullanılabilirliği (usability) ile örtüşmüyor olabilir. Şu soruların cevabı gerekir: Bir sistem 100 fonksiyonlu olarak kullanılabilir mi? Bunun için ne büyüklükte büyük bir menü tasarımı daha uygundur?

Bir başka karşıtlık durumu olarak ürünün düşük maliyeti (low cost) güçlülüğü (robustness) ile çelişebilir. Örnek olarak düşük maliyetli bir sistem, kullanıcı hatalı veri girişi yaptığında hataların (errors) kontrol edilmediği bir çözüm içerebilir.

Geliştirilen sistemin etkinliği (efficiency), taşınabilirliği (portability) ile zıt yönde gerçekleşebilir.

Ürünün modellemesi için tercih edilen hızlı geliştirme (rapid development) yöntemi, geliştirilen sistemin fonksiyonelliği ile çelişebilir. Örneğin bir projenin geliştirilmesi için 5 hafta planlandığı ve bunun 5 geliştirici ile yapılacağı kabul edilsin. Tasarım süresi ise 2 hafta olsun. Bir sorunla karşılaşılması durumunda, ürünün teslim tarihinin uzatılması mümkün değil ise, gerçekleştirilecek fonksiyonellik azaltılacaktır. Bu durumda modeldeki tüm kullanıcı hikayeleri (use cases) implemente edilmeden proje tamamlanacak ya da ürün zamanından sonra teslim edilecektir.

---

[1] Raporun tüm şekilleri bölüm numarası ile başlayarak devam etmelidir. Örneğin Şekil 8.1, Şekil 8.2,….. gibi. Sonraki bölümün şekilleri ise Şekil 9.1, Şekil 9.2, … olarak numaralanacak ve tüm şekillerin yanında mutlaka açıklaması olacaktır.

Hedeflenen ürün maliyeti, yeniden kullanılabilirliğe (reusability) karşıt olan bir özelliktir. Geçmiş yıllarda tasarımın yeniden kullanılabilirliği oldukça güçtü, ekstra çaba gerektiriyordu. Günümüz uygulamalarında tasarım şablonlarıyla yeniden kullanılabilirlik modern bir yapıya oturtularak yaygınlaştı.

Probleminize **uygun olan** tasarım hedefleri aşağıdaki kavramlar içerisinden **seçilerek** problemin çözümü içerisindeki işlevleriyle açıklanacaktır.

Reliability	Modifiability	Maintainability	Understandability	Adaptability
Reusability	Efficiency	Portability	Traceability of requirements	Fault tolerance
Backward-compatibility	Cost-effectiveness	Robustness	High-performance	Good documentation
Well-defined interfaces	User-friendliness	Reuse of components	Rapid development	Minimum number of errors
Ease of learning	Readability	Ease of remembering	Ease of use	Increased productivity
Low-cost	Flexibility	……	……..	……..

9 Önerilen Yazılım Mimarisi

Projenizin yazılım mimarisi probleminizin çözümüne uygun olacak şekilde araştırılmalı ve mimari önerisi açıklanmalıdır.

Görsel betimleme olarak, geliştireceğiniz yazılım ürününün bağlam bakış (context) açısına göre çözümü anlatılacaktır. Bu UML diyagramı "deployment" diyagramdır (https://www.uml-diagrams.org/deployment-diagrams-overview.htm). Burada geliştireceğiniz sistemin alt sistemlere parçalanmasını gösteren fonksiyonel bir model tasarlanmaktadır. Yaptığınız UML çizimi açıklanmalıdır.

10 Sınıf Diyagramları

Bu bölümde mantıksal bakış açısına göre projenin tasarımına devam edilmektedir. Rapor 1 'de "use case" diyagramları ile açıklanan davranışsal betimlemelerin tümüne ait statik etkileşimleri gösteren sınıf diyagramları çizilecektir (https://www.uml-diagrams.org/class-diagrams-overview.html).

Rapor 1 'de eksik olarak tanımlanmış **gereksinimler (kullanıcı hikayeleri) tamamlanmalı,** bu bölümde sınıf diyagramlarının tasarımı olarak eklenmelidir.

"use case" diyagramlarının açıklamalarına eş olarak tanımlanan sınıflar ve sınıflar arasındaki ilişkilerde **"aggregation", composition", "association","generalization/specilization**" ilişkileri mutlaka kullanılmak zorundadır. Bu bağlamda çalışmanın kapsamına / büyüklüğüne uygun sayıda sınıflar arası ilişkinin tanımlanması zorunludur.

Projenin işleyişini tek bir sınıf diyagramında göstermek uygun **DEĞİLDİR**

11 Dinamik Model

Sistemin dinamik modeli (interaction modeling) UML diyagramları ile "collaboration diagrams" ya da "sequence diagrams" (https://www.uml-diagrams.org/communication-diagrams.html) olarak verilebilir.

Çözümünüz bu diyagramların ikisini birden içermez. Çünkü bu diyagramlardan biri diğerinin yerine tercih edilebilir; ikisi de aynı çözümü verirler. Sequence diyagram tercih edilmektedir.

Gruplar projelerine uygun bir dinamik model diyagramı formatı seçer ve çalışmanın büyüklüğüne göre uygun sayıda dinamik model diyagramı tasarlar. Her bir şeklin altında açıklamasının olması **zorunludur.**

Herhangi bir dokümanda açıklaması olmayan bir şeklin/tablonun değeri ve anlamı yoktur.

12 Kullanıcı Arayüzü

Bu bölümde tasarlanacak projenin arayüzlerinin taslakları yapılır. Bir "UI Mockup Tool" kullanılması zorunludur ve hangi tool kullanıldığı mutlaka belirtilmelidir. Kopyala/yapıştır resimler değerlendirmeye alınmaz.

Tasarladığınız ekran görüntüleri **sıralı olarak ve açıklamaları ile** verilir. Bu raporun içeriğindekiler sadece tasarımdır. Bitirme projesinde değişmesi beklenebilir.

UML diyagramlarının tasarımı ile ilgili diğer referanslar:

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-class-diagram/

https://www.uml-diagrams.org/index-references.html
(UML indeksidir, terimlerin tümünün açıklamalarını içerir)

IV Test Planları (16 font) (sayfa başı)

Test dokümantasyonu, yazılımın test edilmesinden önce veya test sırasında oluşturulan çözümlemelerin dokümantasyonudur. Bu doküman test ekibine yardımcı olur. Ekibin testte nelere dikkat etmesi gerektiğine (test effords), testin kapsamına (test coverage), kaynağın izlenmesine (resource tracking), projenin ilerleyişinin tahminine yardımcı olur. Böylece, testin planlaması, testin tasarımı, testin yürütümü, test sonuçlarının tanımlamaları ve testin belgelenmesi test aktivitelerinden çıkarılacaktır. Test dokümantasyonun eksikliğinin ürünün kullanıcı tarafından kabulü (acceptance) için bir sorun olduğu bugün de günceldir.

Özetlemek gerekirse test hazırlıkları, bir yazılım projesinin en önemli parçasıdır ve geliştirilecek projede yapılacak işlemlerin çoğunu ayrıntılı olarak açıklar.

IEEE 829-Test Plan Dokümantasyonunda yazılım testi için

i) neler yapılması gerektiği,
ii) bunların hangi kalite standardına göre yapılacağı,
iii) test işlemlerinin hangi zaman ölçeğinde gerçekleştirileceği,
iv) test prosesini hedeflenenler doğrultusunda sağlamakla alınan riskler ve bunların nasıl gerçekleştirileceği açıklanmaktadır.

"Software Testing Help" web sayfası bu dokümanı niçin ve nasıl hazırlayacağınız konusunda önceki raporlarınızla ilişkili olarak bilgilendirme yapacaktır.
https://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/

13 Test Edilebilecek ve Test Edilemeyecek Özellikler

Bu bölüm, IEEE –829 Test Dokümantasyonu standardı doğrultusunda projenizin alt sistemlerine göre aşağıdaki tablolar formatında hazırlanacaktır. Bunlar geliştirilecek yazılım sisteminde test edilecek alt sistemler ve bu alt sistemlerde test proseslerinin gerçekleştirileceği senaryolardır.

Tablo13.1: …….. Sisteminin Test Edilecek Alt Sistemleri /Özellikleri/….

Test Edilen Alt Sistemler	Test Edilen Senaryolar
Ana Menü Sistemi	Ana menü sistemi kullanılarak oyunun başlatılması veya oyundan çıkılması
Girdi (Input) Sistemi	Kullanıcının kullandığı sisteme göre girdilerinin kontrolü
Oyuncu Aktörünün Sistemi	Oyuncu karakterinin temel mekaniklerinin kullanılması
Düşman Aktörünün Sistemi	Düşman karakterinin temel komutlarının gerçekleştirilmesi
Oyun Ses Sistemi	Oyunda bulunan farklı seslerin kontrolleri ve etkileşimi
Arayüz Sistemi	Oyunda olan oyuncunun gördüğü değişen arayüz etkenlerinin kontrolü

Aşağıdaki tablo ise geliştirilecek yazılım sistemin test işlemine sokmayacağı özellikleri özetlemektedir. Bu alt sistemlerin ilgili özelliklerinin niçin test işlemine sokulmadığı mutlaka açıklanmalıdır.

Tablo 13.2: …….. Sisteminin Test Edilmeyecek Alt Sistemleri/Özellikleri/….

Test Edilemeyen Alt Sistemler	Test Edilemeyen Özellikler
Arayüz Farklılık Sistemi	Arayüzün farklı platformlardaki değişkenleri
Kontrol Farklılıkları Sistemi	Oyuncunun kullandığı kontrol cihazı tipine göre yaşanabilecek olan girdi gecikmeleri

Test edilecek ve edilmeyecek özelliklerin tümü belirlendikten ve kısaca açıklaması yapıldıktan sonra
test prosesinin bir bütün olarak nasıl gerçekleştirileceği açıklanır. Bunlar aslında testin planlanması
aşamasında hedeflenmiş olan *test düzeyleri*, test tipleri ve *test yöntemleri* olarak:
i)Testin el ile ya da otomatik gerçekleştirileceği
ii) Beyaz kutu /kara kutu/ gri kutu testlerinden hangilerinin yapılacağıdır.

El ile gerçekleştirilecek testleri mutlaka tanımlanmalıdır.

Tablo 13.3: Elle gerçekleştirilecek testlerin açıklaması

Elle Gerçekleştirilen Test	Test Edilen Özellik	Test Türü
Oyunu başlatmak için "Oyunu Başlat" tuşuna bas	Kullanıcının yeni oyunu başlatması	Fonksiyonellik/Kullanılabilirlik
Oyunu başlatmak için "Oyuna Devam Et" butonuna bas	Kullanıcının kayıtlı oyuna devam etmesi	Fonksiyonellik/Kullanılabilirlik
Oyundan çıkmak için "Oyundan Çık" Tuşuna bas	Kullanıcının oyundan başarılı bir şekilde çıkması	Fonksiyonellik/Kullanılabilirlik
Karakteri hareket ettirmek için atanmış olan yön tuşlarına bas	Kullanıcının karakteri hareket ettirmesi	Fonksiyonellik/Kullanılabilirlik
Karakterin saldırı aksiyonunu alması için atanmış olan saldırı tuşlarına bas	Kullanıcının, karakterin saldırıya geçmesini sağlaması	Fonksiyonellik/Kullanılabilirlik
Karakterin kaçınması için tanımlanmış olan tuşa bas	Kullanıcının, karakterin kaçınma aksiyonunu gerçekleştirmesini sağlaması	Fonksiyonellik/Kullanılabilirlik
Karakteri oyun içerisinde aşılmaması hedeflenen duvara yönlendir	Karakterin oyun içerisindeki engel ile etkileşimi	Fonksiyonellik/Kullanılabilirlik
Karakteri, kullanıcı olmayan düşmana yönlerdir	Karakterin oyun içerisindeki kullanıcı olmayan düşmanlar ile etkileşimi	Fonksiyonellik/Kullanılabilirlik
Karakteri, kullanıcı olmayan karakterle konuştur	Karakterin, kullanıcı olmayan karakterler ile konuşması	Fonksiyonellik/Kullanılabilirlik
Karakteri, kullanıcı olmayan karakterle, işlemi gerçekleştirecek iletişimde bulundur	Karakterin, kullanıcı olmayan karakterle olan eşya değiş tokuşu gerçekleştirmesi	Fonksiyonellik/Kullanılabilirlik

14 Test Cases

Sistemin alt sistemlerine hangi testler uygulanacak ise her birinin bir test kimliği (test case id) olmalıdır. Her bir alt sistemin özelliklerine (features) hangi testin uygulanacağına (functional /unit/integration veya diğerleri) karar verilir.

Her bir *test case* için input specification (giriş betimlemeleri) ya data (veri) olarak alfanumerik değerler ya da tıklama ya da herhangi bir seçim yapma gibi kişisel eylemlerdir.

Her bir *test case* için beklenen çıktılar (expected outcomes) hatanın (error) kaynağı, hatanın önem derecesi (error severity) ya da hata tipleri (types of error) şeklinde elde edilecektir

Ayrıntılı Test Durumları ve Sonuçları (tablo olarak da hazırlanabilir)

Test Case ID: TCH-01

Test Amaçları: Kullanıcının sisteme başarılı bir şekilde kayıt olması ve giriş yapması kontrolü.

Test Prosedürleri ve Sonuçları: Kullanıcının sistem tarafından istenen bilgileri eksiksiz doldurup kayıt olması veya giriş yapması

Test Verisi: Veri

Beklenen Sonuç: Kullanıcının kayıt için istenen verileri eksiksiz bir şekilde doldurduktan sonra uygulamanın verileri veri tabanına sorunsuz kaydetmesi veya giriş yapabilmesi

Gerçek Sonuç: Beklenen sonuç ile aynı.

Hata Türü: Eksik veya yanlış veri / Bağlantı hatası

Hata Önemi: Önemli

Test Case ID: TCH-02

Test Amaçları: Owner'ın sistemde başarılı bir şekilde Property oluşturması.

Test Prosedürleri ve Sonuçları: Owner'ın sistem tarafından istenen bilgileri eksiksiz doldurması.

Test Verisi: Veri

Beklenen Sonuç: Owner Propert oluşturmak için istenen verileri eksiksiz bir şekilde doldurduktan sonra uygulamanın verileri veri tabanına sorunsuz kaydetmesi

Gerçek Sonuç: Beklenen sonuç ile aynı.

Hata Türü: Eksik veya yanlış veri / Bağlantı hatası

Hata Önemi: Önemli

Test Case ID: TCH-03

Test Amaçları: Owner'ın sistemde Admin için hesap oluşturması oluşturulması.

Test Prosedürleri ve Sonuçları: Owner'ın sistem tarafından istenen bilgileri eksiksiz doldurması.

Test Verisi: Veri

Beklenen Sonuç: Owner Admin hesabı oluşturmak için istenen verileri eksiksiz bir şekilde doldurduktan sonra uygulamanın verileri veri tabanına sorunsuz kaydetmesi.

Gerçek Sonuç: Beklenen sonuç ile aynı.

Hata Türü: Eksik veya yanlış veri / Bağlantı hatası Hata Önemi: Önemli

……..
…….
……...…

Çalışma grubunun (grup sayısının büyüklüğü ile orantılı) gerçekleştirdikleri aktivitelere(eylemlere) paralel olarak gerçekleştirecekleri testler de benzer büyüklükte olmalıdır. Bu nedenle önceki raporlarda gerçekleştirilen eylemlere bağlı olarak test durumları tanımlamalı ve her bir "test case" için kimlikleri de belirtmelidir.

Not: Verilen örnekler bu bölümdeki tasarımınızla ilgili olarak kesinlikle bağlayıcı değildir.

V Sözlük (Sayfa Başı)

Yazılım Mühendisliği Tasarım Projesi raporlarının tümü içerisinde bu bölümde bulunmasına karar verilen önemli kavramlar alfabetik olarak sıralanacak ve birkaç satırlık kısa tanımlamaları yapılacaktır. 8-10 tane kavramın açıklanması beklenir. Bunlar, vize ve final tasarım projesi raporlarından eşit olarak seçilmelidir.