Beykent Üniversitesi
Yazılım Mühendisliği Bölümü
Yazılım Mühendisliği Tasarım Projesi

Rapor 2
2021 – GÜZ

Proje İsmi (kısaltması ile birlikte)

Grup Elemanları
No, Adı Soyadı (Soyadına Göre Alfabetik Sırada)

III Tasarım Dokümantasyonu (16 font)

9 Tasarım Hedeflerinin Tanımlanması (14 font)

Tasarım amaçları (hedefleri), çözümü gerçekleştirilecek sistemin en önemli özelliklerini içerir ve sistemin genel tasarımında etkisi çoktur. Örneğin, bilgisayar oyunlarında öncelik, çözümün doğruluğundan (accuracy) ziyade hızının çok yüksek olmasıdır. Bu nedenle de, bir bilgisayar oyununda kullanılan makine (physical engine) , oyunun daha hızlı çalışmasını sağlayacak varsayımlarda bulunabilir. Burada hedef sistemin doğruluğu (accuracy) olmaz. Ama, başka bir örnekte örneğin önemli bilimsel hesaplamaların yapıldığı bir projede fizik hesaplamaları, hız pahasına bile olsa, en önemli kriter olarak, ne şekilde olursa olsun, doğru hesaplanmalıdır.

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.

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.

Sistem tasarımında probleme uygun bazı kavramların birbirinin yerine geçebileceğinin (trade-offs) belirlenmesi önemlidir. Örneğin:

i) fonksiyonellik (functionality) kullanılabilirliğe (usability) karşıttır. Bir sistem 100 fonksiyonlu olarak kullanılabilir mi? Bunun için ne büyüklükte büyük bir menü tasarımı daha uygundur?

ii) Düşük maliyet, (low cost) güçlülük (robustness) ile çelişir. Düşük maliyetli bir sistem, kullanıcı hatalı veri girişi yaptığında hataları (errors) kontrol etmemiş olabilir.

iii) Etkinlik(Efficiency), taşınabilirliğe(portability) karşıdır.

iv) Hızlı geliştirme (rapid development), fonksiyonellik ile çelişir. Örneğin bir projeye geliştirme için 5 hafta verildiği ve bunun 5 programcı ile yapılacağı kabul edilsin. Tasarım süresi ise 2 hafta olsun. Bir sorun durumunda, teslim tarihinin uzatılması mümkün değil ise, fonksiyonellik azaltılır. Bu durumda modeldeki tüm “use cases” (kullanım durumları) uygulanmayacak veya proje teslim edilmeyecektir.

v) Maliyet (cost), yeniden kullanılabilirliğe (reusability) karşıt olan bir özelliktir. Geçmişte, tasarımın yeniden kullanılabilir hale getirilmesi için ekstra çaba gerekiyordu. Kodlamada nesneler arasındaki pek çok özelliğin yeniden tasarımı yapılmalı idi. Günümüzde, tasarım şablonları ile bu değişim de oldukça modernleşti.

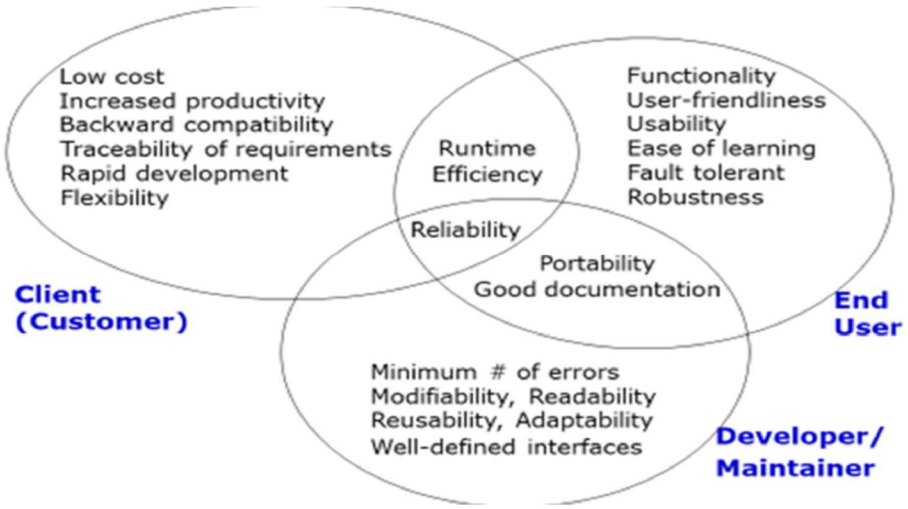
Tasarım şablonları kullanılarak tekrar kullanılabilirlik oldukça ucuzlatılabilmektedir.

vi) Bu örnekler çoğaltılabilir.

Probleminize ait tasarım amaçları örnekleri olarak aşağıdakiler alınabilir. Tasarım amaçlarınızı belirledikten sonra bunların probleminize uygun olarak ifade edilmesi gerekmektedir.

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	……	……..	……..

Projeye katkı sağlayan herkesin (stakeholders) tasarım amaçlarından farklı beklentileri vardır. Bunların arakesitindeki hedefleri belirlemek her proje için önemlidir. Aşağıdaki alıntı (Object Oriented Software Engineering , Using UML , Patterns and Java – Bernd Bruegge & Allen H. Dutoit) gerek tasarım hedeflerinin belirlenmesinde, gerekse özelliklerin birbirinin yerine geçtiği cümlelerin oluşturulmasına yardımcı olacaktır.



10 Önerilen Yazılım Mimarisi

Bu bölümün açıklaması projenize uygun olarak ayrıntılı olarak araştırılmalı ve yazılı olarak açıklanmalıdır. Görsel betimlemede UML diyagramı olarak "package" diyagramın kullanılması uygun olabilir (https://www.uml-diagrams.org/package-diagrams-overview.html)

Daha sonra bağlam bakış (context) açısına göre çözüm için önerilen tasarım özetlenir. Bu da "deployment" diyagramın ayrıntılı açıklamasına karşılık gelecektir.

11 Sınıf Diyagramları

Önceki raporda "use case" diyagramları ile açıklanan davranışsal betimlemelerin tümüne ait statik etkileşimlerini gösteren sınıf diyagramları çizilecektir (https://www.uml-diagrams.org/class-diagrams-overview.html). Bu bölümde mantıksal bakış açısına göre projenin tasarımına devam edilir; varlıklar (entities) sınıflar olarak tasarlandığında UML sınıf ve nesne diyagramları tasarlanmış olacaktır.

"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.

12 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. Raporda bu diyagramların ikisi de yer almamalıdır. Zire bu diyagramlardan biri diğerinin yerine tasarlanabilir, aynı çözümü verirler. 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.** Bir raporda açıklaması olmayan herhangi bir şeklin anlamı yoktur.

13 Altsistem Ayrıştırması (Subsystem Decomposition)

Geliştirecek sistemin alt sistemlere parçalanması gösteren fonksiyonel modeldir. Bunlar "layers and partitions, system information flow (topology)" olarak verilir. Sistem ayrıştırması (decomposition) UML deployment diyagram ile tasarlanabilir (https://www.uml-diagrams.org/deployment-diagrams-overview.html).

14 Kullanıcı Arayüzü

Bu bölümde tasarlanacak projenin arayüzlerinin taslakları yapılır. Bir "IU 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.

Not: UML diyagramlarının tasarımı referansı: https://www.uml-diagrams.org/index-references.html