Yazılım İnşasına Özet Bir Giriş:

Sedat Görmüş, PhD

Email : sedatgormus@gmail.com

: sedatgormus@ceng.ktu.edu.tr

İçerik

- Yazılım Geliştiriciliği Nedir?
- Yazılım İnşası Bir Yazılımcı İçin Neden Önemli?
- Yazılım İnşasında Metaforlar(Mecazlar)
- Metaforları Nasıl Kullanırız?
- Genel Yazılım Metaforları

Yazılım Geliştiriciliği Nedir?

- Yazılım geliştiriciliği problemin karmaşıklığına göre çok karmaşık bir problem olabilir.
- Yazılım geliştirme bir problemi bilgisayar yoluyla kodlamaktan daha öte adımlar içrerir.
- Son yıllarda, geliştiriciler, geliştirme aktivitesinin başarısını arttırmak için belli başlı geliştirme aktiviteleri belirlediler:
  - **Problem tanımlama** (Problem definition)
  - **Problemi çözmek için gereksinim belirlenmesi** (Requirements)
  - **Yazılımı geliştirme için gerekli adımları planlama** (Construction Planning)
  - **Yazılım mimarisinin taslağının çıkarılması** (High level software architecture)
  - **Detaylı tasarım** (Detailed Design)
  - **Kodlama ve Hata ayıklama** (Coding and Debugging)
  - **Parça testi** (Unit Testing)
  - **Birleştirme Testi** (Integration testing)
  - **Birleştirme** (Integration)
  - **Sistem Testi** (System Testing)
  - **Bakım ve hata giderme** (Corrective Maintenance)

Neden Proje Geliştirme Adımlarını İzlemeliyiz

Table 3-1. Average Cost of Fixing Defects Based on When They're Introduced and Detected

	Time Detected				
Time Introduced	**Requirements**	**Architecture**	**Construction**	**System Test**	**Post-Release**
Requirements	1	3	5–10	10	10–100
Architecture	—	1	10	15	25–100
Construction	—	—	1	10	10–25

Source: Adapted from "Design and Code Inspections to Reduce Errors in Program Development" (Fagan 1976), *Software Defect Removal* (Dunn 1984), "Software Process Improvement at Hughes Aircraft" (Humphrey, Snyder, and Willis 1991), "Calculating the Return on Investment from More

Problemi Doğru Tanımlama Çözüm İçin Vazgeçilmez İlk Adımdır

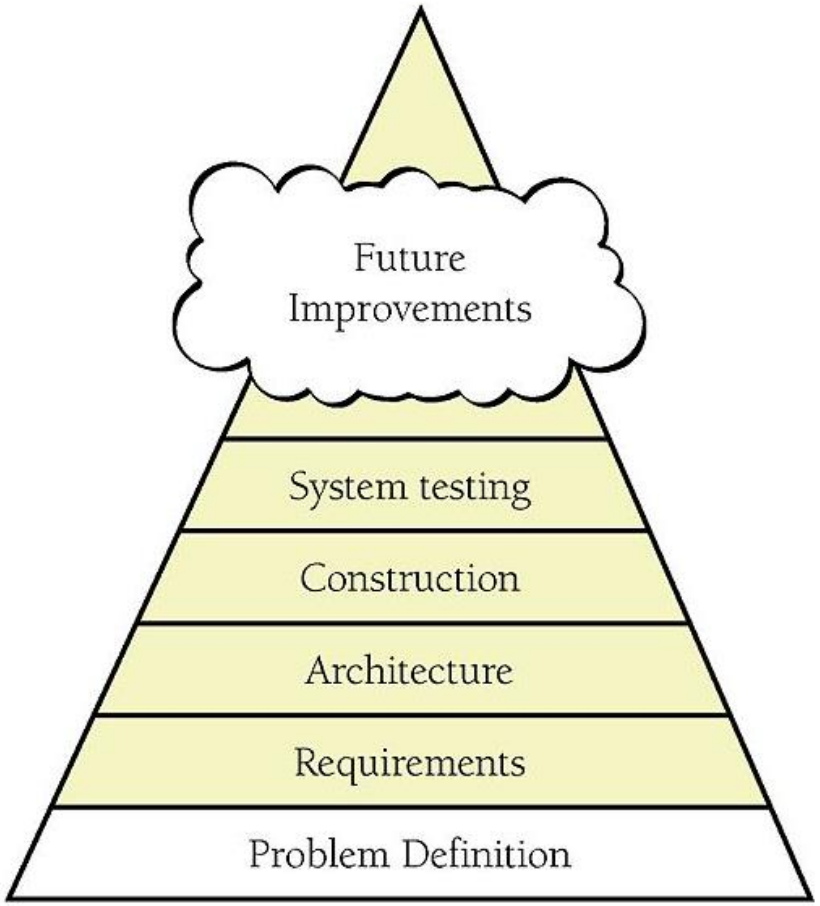
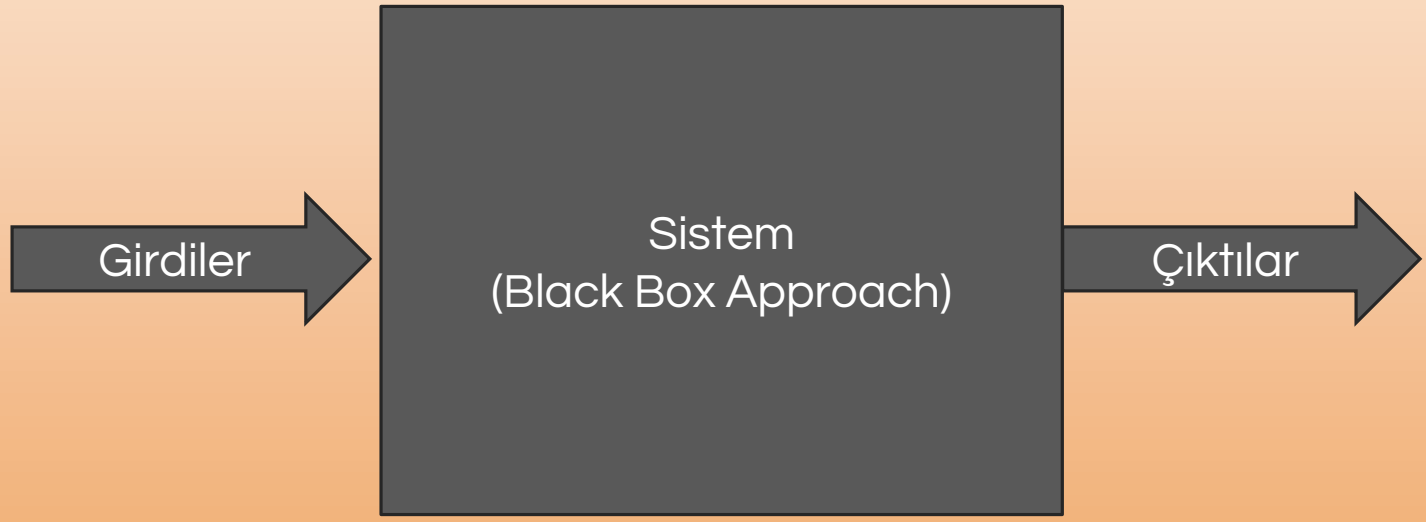


Figure 3-4. The problem definition lays the foundation for the rest of the programming process

Problem Tanımlayalım

- **Şimdi Siz MegaMart Web uygulamasından sorumlu geliştiricisiniz.**
- **Aşağıdaki İfadelerden hangisi sizin için doğru bir problem tanımı olur**
  - MegaMart Web uygulaması gelen siparişleri optimum şekilde karşılayamıyor.
  - *SuperMart'ın gelen siparişleri karşılaması için, web uygulamasındaki gömülü sunucu tabanlı sınıfların optimizasyon probleminin çözülmesi gerekir.*

Gereksinimler



Gereksinimler

- Gereksinimler detaylı olarak yazılımın **yapması gereken fonksiyonları** sıralar.
- Gereksinim analizinin ana hedefi **proje risklerini** azaltmaktır.
- Gereksinimlerin doğru tanımlandığını aşağıdaki sorularla kontrol edebilirsiniz.
  - Sistem girdileri belirlendi mi?
    - Sistem girdilerinin kaynakları belirlendi mi?
    - Sistem girdilerinin kararlılığı ve doğruluğu test edildi mi?
    - Sistem girdilerinin değer aralıkları belirlendi mi?
    - Sistem girdilerinin sıklığı belirlendi mi?
  - Sistem çıktıları kararlılıkları, hedefleri, veri aralıkları, sıklıkları ve formatları dahil olmak üzere belirlendi mi?
  - Çıktı formatlarının farklı hedefler için belirlenmesi yapıldı mı?(Web sayfaları raporlar, vesaire)

Gereksinim kontrol soruları devam

- Dış donanım ve yazılım arayüzleri eksiksiz bir şekilde belirlendi mi?
- Dış haberleşme arayüzleri belirlendi mi?(handshaking, error-checking, ve haberleşme protokolleri dahil olarak.)
- Kullanıcının gereksinimi olan bütün görevler belirlendi mi?
- Her görev (task) için gerekli veri ve her görevden çıkacak veriler belirlendi mi?
- Uygulamanın tepki süresi kullanıcın beklentisiyle bütün parçalar ve operasyonlar için örtüşüyor mu?
- Diğer zamanlama kısıtlamaları belirlendi mi?
  - Örneğin veri transfer zamanı, hesaplama zamanı, ve sistem transfer hızı?

Gereksinim kontrol soruları devam

- Sistem için gerekli güvenlik prosedürleri ve seviyeleri belirlendi mi?
- Sistemden beklenen kararlılık ve ayakta kalma süresi (up-time) belirlendi mi?
- Ayrıca, eğer sistemde bir arıza olursa verilerin sağlıklı şekilde saklanması garati edilebilir mi?
- Makineler için hafıza ve disk kapasiteleri belirlendi mi?
- Sistemin bakım yapılabilirliği ve farklı fonksiyonel değişikliklere uygunluğu, çevresel faktörlerdeki değişime uygunluğu, yazılımdaki arayüzlerde olacak değişimlere uygunluğu belirlendi mi?*
- Başarı ve başarısızlığın tanımı ve projeden minimum beklentiler belirlendi mi?

Gereksinimlerin *Kalitesinin* Denetlenmesi

- Gereksinimler kullanıcının dilinde, kolayca anlaşılır biçimde mi yazıldı?
- Gereksinimler birbiriyle çelişiyor mu?
- Birbirleriyle ilintili sistem çıktıları arasındaki tradeofflar belirlendi mi? (Örneğin, kararlılık ve doğruluk)
- Gereksinimler dizayn ayağına dokunmuyor mu ? (gereksinim dizayn içermmemeli)
- Gereksinimler yeterli detaylara sahip mi?
  - Herhangi bir gereksinim daha detaylandırılmalı mı?
  - Herhangi bir gereksinim gereksiz yere detaylandırılmış mı?

Gereksinimlerin *Kalitesinin* Denetlenmesi

- Her şeyden önemlisi, gereksinimler herhangi bir geliştirici grubuna verildiğinde onlar tarafından ***kolayca anlaşılıp, dizayn aşamasına geçilebilir mi?***
- Gereksinim parçaları problem için ***gerekli mi ve gereksinimden orijinal probleme varılabilir mi*** ? (Bazen sorunla alakasız gereksinimler listeye eklenebilir)
- Her gereksinim sistem içinde ***bağımsız olarak test*** edilebilir mi?
- Eğer gereksinimler değişecekse, ***değişme ihtimali olan gereksinimler*** belirlendi mi? Ve nasıl değişebilecekleri belirlendi mi?

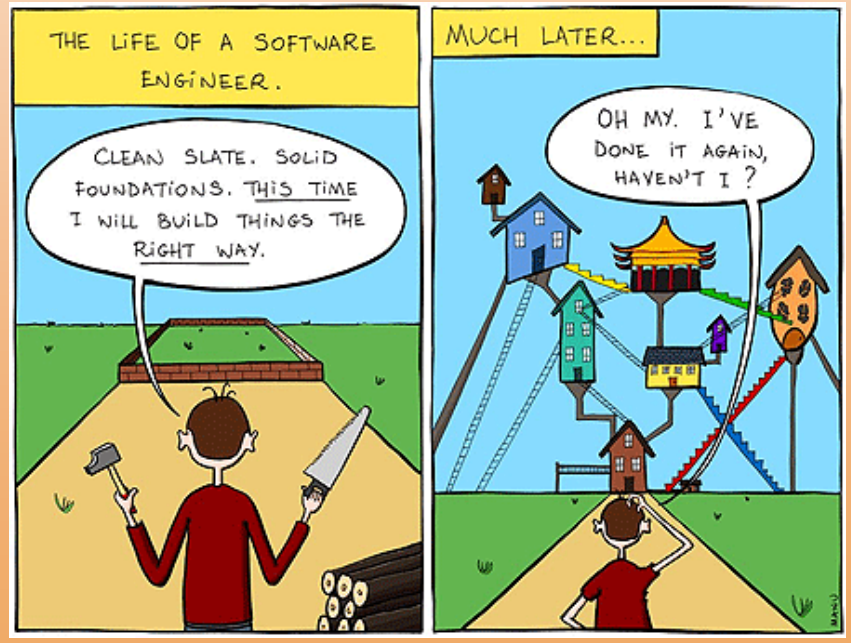
Ve son olarak; Gereksinimler tam mıdır?

- Eğer gereksinim sırasında bazı alanlar tanımlanmamışsa, ***tanımlanmamış alanlar belirlendi*** mi?
- Gereksinimler sağlandığında ürün kullanıcı tarafından ***kabul edilebilir*** bulunacak mıdır?
- Uygulaması zor yada ***imkansız gereksinimleri*** ayıkladınız mı? Ya da uygulanabilir hale getirdiniz mi?
- Unutmayalım, gereksinim analizinin amacı bizim için projedeki riskleri azaltmaktır.

Gereksinim belirleme calışması

Şimdi sizden 5 kişilik gruplar oluşturarak bahçe bakımı yapmak için online bahçıvanları bulabileceğiniz bir servis için gereksinim analizi yapmanızı istiyorum. Lütfen gereksinimlerinizi yukarıdaki gereksinim kontrol listesine göre değerlendiriniz. 20 dakika sonra grubun liderinden gereksinimleri ve gereksinimlerin kontrol listesiyle olan uyumunu dinleyeceğiz.

Mimari Tasarım



Yazılım Mimarisi

- Mimari, yazılımın ***üst seviyede*** dizayn edilmesidir.
- Mimarinin ***kalitesi, yazılımın*** kalitesini direkt olarak etkileyecetir.
- Mimari, yazılımcılara gerekli yazılım ***şablonlarını oluşturmaları*** için yardımcı olacaktır.
- Mimari ***programlanacak sistem parcalarını*** açıkça ifade ettiği için, yazılım geliştiricileri farklı sitem parçaları üzerinde ***paralel olarak*** çalışabilirler.
- İyi bir mimari yazılım inşasını ***kolaylaştırır***, ancak kötü mimari yazılımın inşasını neredeyse ***imkansız*** hale getirir.

TİPİK MİMARİ PARÇALARI

- Mimari projenin ***ana yapı taşlarını*** tanımlamalıdır.
- Projenin çapına göre, her yapı taşı ***bir sınıf olabileceği*** gibi, bir den ***fazla sınıfı da içerebilir***.
- Her yapı taşını oluşturan sınıf yada sınıflar birlikte çalışan bir ***üst seviye fonksiyonu*** yerine getirir (Örneğin, ***kullanıcıyla etkileşim***, ***Web sayfalarını gösterme***, ***komutları yorumlama***, v.s)
- Gereksinim listesindeki her gereksinim en ***az bir yapı taşı*** tarafından kapsanmalıdır.

TİPİK MİMARİ PARÇALARI

- Her yapı taşının fonksiyonu ***hiçbir süpheye*** yer bırakmakzsızın çok iyi şekilde tanımlanmalıdır.
- Ve her yapı taşı diğer yapı taşıyla ilgili mümkün olduğunca ***az bilgi içermelidir ve bilmelidir***.
- Bloklar arasındaki haberleşme kuralları ***çok iyi şekilde*** tanımlanmalıdır.
- Mimari hangi yapı taşının diğer yapı taşlarını kullanabileceğini ***açıkça ifade etmelidir***.

Ana Sınıflar

- Mimari ***Ana sınıfları*** tanımlamalıdır.
- Mimari bu sınıfların ***sorumluluklarını*** tanımlamalıdır.
- Ayrıca, mimari bu ana sınıflar arasındaki ***etkileşimi*** de belirlemelidir.
- Mimari sınıflar arasındaki ***hiyerarşiyi*** belirlemelidir.

Veri(tabanı) Dizaynı

- Mimari ana ***dosya*** ve ***tablo*** yapılarının ***dizaynlarını*** içermelidir. İnşa sırasında yazılım geliştiriciler, mimariyi dizayn ve gereksinimleri bu şekilde kolayca anlayabilirler.
- Ayrıca, ***bakım*** sırasında mühendislerin işlerini yapması ***kolaylaşır***.
- Veri sadece bir alt sistem yada sınıf tarafından ***erişilmeli ve kontrol*** edilmelidir. (Mesela bir veritabanı uygulamasında ***veri tabanı işlemlerini bir sınıf yada alt sistem*** yapmalıdır)
- Mimari veri tabanlarının ***üst seviye yapılarını*** vermelidir.

Bussiness Rules(İş kuralları)

- Eğer mimari bazı ‘bussiness rules’ gerektiriyorsa, onları belirlemeli ve sistem ***dizaynına olan etkilerini*** tanımlamalıdır.

- Örneğin dizayn edilen sistem gelen verilerin asla ***3 saniyeden daha fazla tamponlanmamasını*** gerektiriyorsa (near realtime – yada realtime), aşağıdaki sistem parçaları bu kurallara göre dizayn edilecektir.
  - ***Yazılım parçaları.***
  - ***Veri transferini yapacak teknoloji donanım ve protokoller.***

Kullanıcı Arayüzü (User Interface)

- Kullanıcı arayüzü çoğu zaman ***gereksinim aşamasında*** tanımlanır.
- Eğer tanımlanmamışsa, ***mimari aşamasında dizayn*** edilmelidir.
- Bu aşamada ana ***GUI elemanları, Web sayfaları, v.s. Tanımlanmalı ve dizayn*** edilmelidir.
- Kullanıcı arayüzü dizaynı, bir programın ***kullanılması yada çöpe*** gitmesi arasındaki ince çizgidir.

Kaynak Yönetimi

- Mimari ***kaynak yönetimiyle*** ilgili planları içermelidir.

- Kaynaklardan bazıları
  - ***Hafıza (özelikle sürücü dizaynı ve gömülü sistemler)***
  - ***CPU***
  - ***Disk***
  - ***İletişim kaynakları***
  - ***FPGA içindeki kapı miktarı***
  - …
- Özelikler karmaşık sistemlerde ***ölçeklenebilirlik*** önemli bir faktör olarak karşımıza çıkmaktadır

Performans

- Performans beklentileri ***gereksinimlerde*** tanımlanmalıdır.
- Mimari ise performans ***parametrelerinin*** birer tahminini içermelidir.
- Eğer bazı alanlarda ***performans sağlanmama riski*** varsa mimari bunu belirtmelidir. (Bu sayede geliştiriciler, bu alana daha fazla özen gösterebilirler)
- Ayrıca mimari her sınıf için ***geliştirme süresi gereksinimlerini*** içermelidir.

Lokalizasyon (Yerelleştirme)

- Eğer yapılan proje farklı ***coğrafyalarda çalışacaksa*****,** birden farklı dili içermelidir.
- Bunun için **mimari farklı ortamları desteklemek** için ***gerekli mekanizmaları*** tanımlamalıdır.
- Arayüz ***dili, hata mesajları, log mesajları***, v.s. Birden fazla dilde tanımlanmalıdır.
- **Soru: Farklı dilleri desteklemek için uygun bir mimari tasarlanmanız istenirse, nasıl bir yöntem aklınıza gelir?**

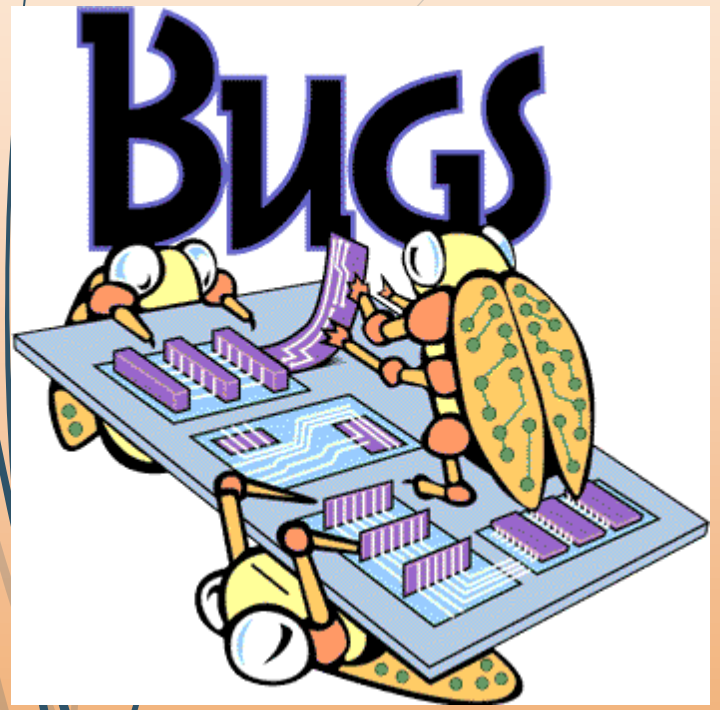
Hata İşlemleri

- Hata ***ayıklama ve işleme***, bir çok proje için **en büyük problemlerden birini** teşkil eder.

- Dolayısıyla, hata ***ayıklama ve yakalama*** için yazılacak rutinlerin nasıl dizayn edileceği mimarinin bir parçası olmalıdır. (**Hata yakalama alışıla gelmiş olarak kodun içinde yapılan bir uygulamadır**)

- Hata ayıklama için mimari dizaynı sırasında sorulacak bazı sorular:
  - **Hata işlemleri**, ***düzelticimi yoksa hatayı bulmak için mi yapılıyor***?
  - Eğer **düzeltici** ise, hatadan ***sağlıklı bir şekilde*** kurtulabilir miyiz?
  - Eğer hata **yakalama** için yapılıyorsa, program ***hatayı log*** dosyasına yazıp, bir şey olmamış gibi devam edebilir yada ***çıkabilir***.
  - Her iki durumda kullanıcıya bir ***uyarı*** göndermelidir.
  - **Program** hata mesajlarını ***sistemde*** nasıl dağıtmalıdır (propagate)?

Hata işlemleri



- Hata mesajları ***çıktı formatları ve hata ayıklama yöntemleri ve kod stilleri ve yönelimleri belirlendi*** mi?
  - Karma karışık bir hata ayıklama yaklaşımı, kodu ve hata ayıklama sürecini ***içinden çıkılmaz hale getirir***. (Güzel ve standart kod yaklaşımı yazılımın her parçası için geçerlidir).
- Hatalar nasıl işlenecek?
  - Mimari yazılımın hangi durumlarda ***hata vereceğini ve bu hataların nasıl yakalanacağını*** belirlemelidir.
  - Örneğin, verilen hatalar hata ***ayıklama sınıfına mı yönlendirilecek***?
  - Yada yazılımda oluşan hata hatayı oluşturan ***ilk komutun olduğu noktaya kadar yönlendirilecek*** mi?
- Sınıfların ***gelen verilerin geçerliliğini*** belirlemek için sorumluluğu nedir?
  - Her sınıf kendi verisini ***kontrol etmeli midir yada bir sistem sınıfı*** mı bu işlemi yapacaktır?
  - Sınıflar ***gelen verinin temiz olduğunu*** varsayabilirler mi?
  - Hata ayıklama sistemini ***siz mi dizayn edeceksiniz*** yada ***programlama dilinin hata ayıklama sistemini m***i kullanacaksınız?

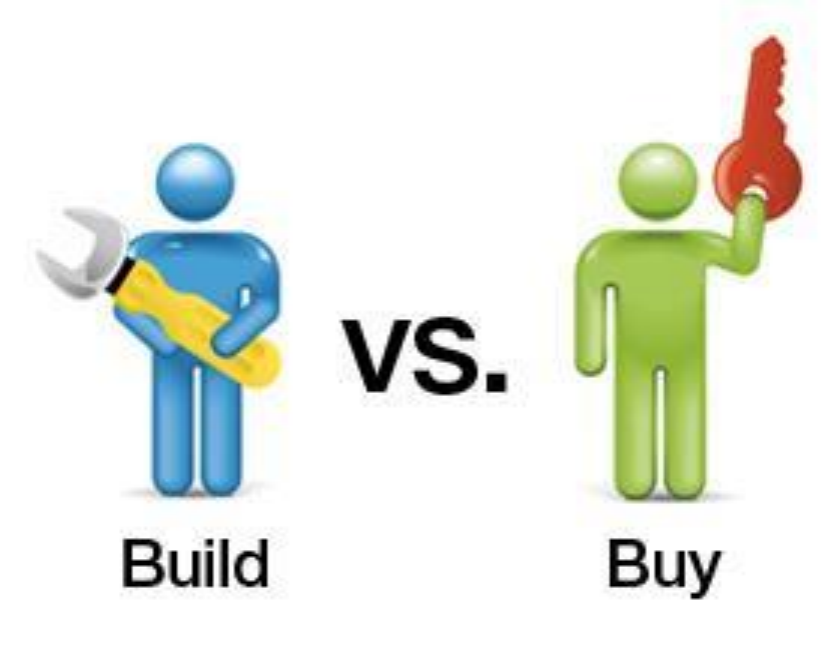
Dayanıklılık(Fault Tolerance)

- Dayanıklılık(fault tolerance) ***sistemin kararlılığını arttırmak*** için kullanılan teknikler bütünü olarak görülebilir.
- Mimari projeden beklenen ***dayanıklılık şeklini*** ve ***hedefini*** belirlemelidir.
- Örneğin, bir sistem bir sayının karekökünü farklı dayanıklılık ilkeriyle şu şekilde hesaplayabilir;
  - System bir ***hata algıladığında,*** hata olmadan hesap yapılan en ***son noktaya*** geri dönebilir.
  - Sistem hata algıladığında, ***yedekte tutulan bir kod*** parçasını koşabilir.
  - Sistem birden farklı ***kare kök alma sınıfını*** içinde barındırır ve sistem he sınıfın ***çıktısını diğerleriyle karşılaştırır.*** Ve sonuç ***bir oylama yöntemiyle*** belirlenir.

Mimari Fizibilite

- Mimariyi dizayn eden grup, ***sistemim performans*** hedeflerini yakalayıp yakalamayacağından emin olmalıdır.
- Bu kaygılar;
  - ***Kaynak gereksinimleri***
  - ***Performans hedefleri***
  - Ve ***geliştirme ortamı*** tarafından destek yönleriyle incelenmelidir,
- Tasarım yapan grup sistemin t***eknik olarak mümkün ve gerçeklenebilir*** olduğunu göstermelidir.
- Her fizibilite ***risk faktörü*** tasarım yapan kişi(ler) tarafından belirtilmeli, ve bu risklerin ***nasıl engellenebileceği*** belirtilmelidir.

Satın alma vs. Programlama



- Proje parçalarının tamamının aynı grup tarafından eksiksiz ***kodlanması***, bazı projeler için ***pahalı*** olabilir.
- Bir çok başarılı proje ***açık ve ücretsiz kaynak kodlu*** projelerden türemiştir.
- Bu yüzden projenin parçalarını mimaride ***satın alınacak ve kodlanacak olmak üzere*** ayırmak da mimari dizaynın bir parçası olmalıdır.
- Eğer proje ***hazır komponentleri kullanıyorsa***, bu komponentlerin ***API ve arayüzleri mimari dizayna dahil*** edilmelidir.

Tekrar Kullanma ve Mimari (kod) değiştirme Stratejisi

- Eğer projede ***daha önce yazılmış*** kodlar kullanılacaksa, bu kodların ***yeni projede nasıl kullanılacağı*** mimari dizaynda belirtilmelidir.
- Daha önce belirttiğimiz gibi ***gereksinimler ve mimari dizayn*** proje sırasında değişecektir.
  - Bu durumda bir ***değiştirme stratejisi*** belirlenmelidir.
  - Ayrıca mimari olabildiğince ***esnek*** yapılarak değişikliklerin projeye olan etkisi azaltılmalıdır.
  - Mimari dizaynı yapan grup, ***olası değişiklikleri*** listelemeli, ve ***nasıl gerçekleşeceklerini ve sisteme etkilerini*** sıralamalıdır.

UML tabanlı Para Oyunu Tasarımı

UML Use Case I

Coin Flip,

Use Case ifadesi

1) Random seçilen bir oyuncu yazı ya da tura olmak üzere bir seçim yapar. Diğer oyuncu ise otomatik olarak diğer seçimi alır. Para atılır. Kazanan oyuncu ilan edilir.

Tetikleyiciler

1) Random seçilen bir oyuncu bir yazı/tura tahmini yapar.

Aktörler

1) Tahmin eden oyuncu
2) Diğer oyuncu
3) Para
4) Oyun

Ön Koşullar

1) İki oyuncu
2) Bir para

Hedefler

1) Bir oyuncu kazanır, diğeri kaybeder

Use-Case'de olup burada olmayanlar

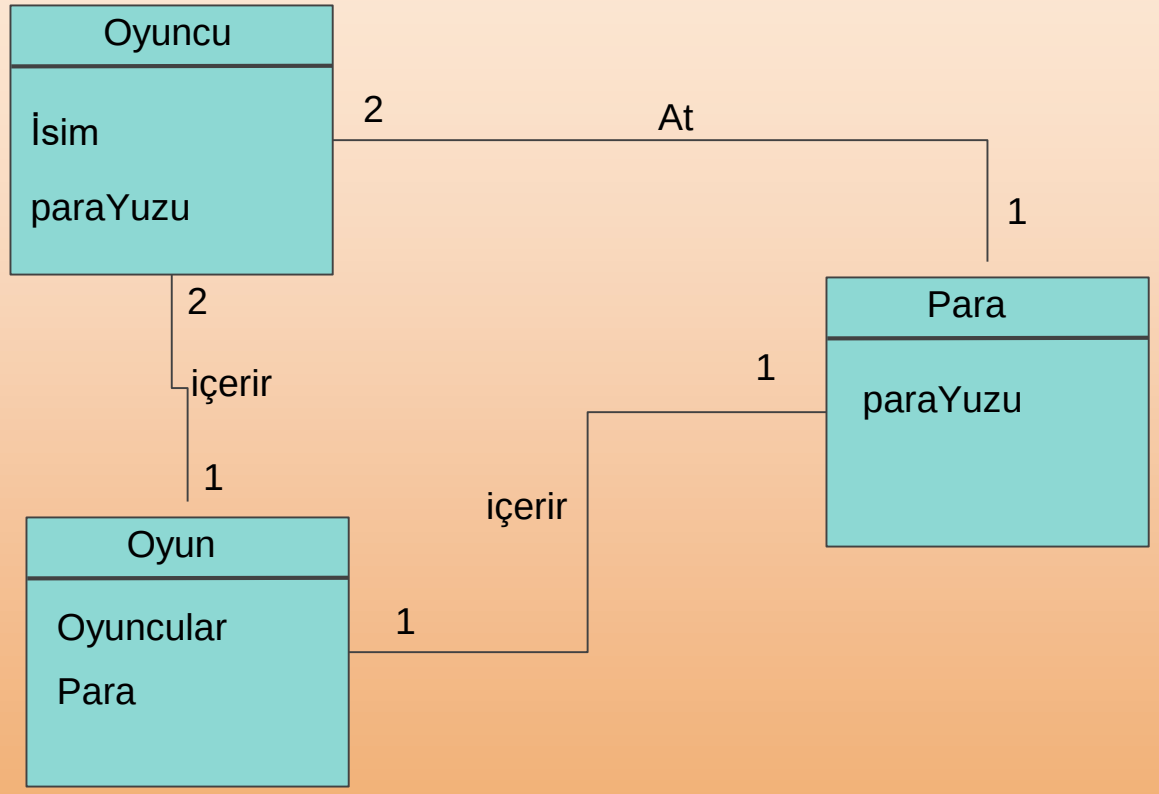
1) Calışmazsa ne olur.
2) Alternatifler nelerdir

UML Use Case II

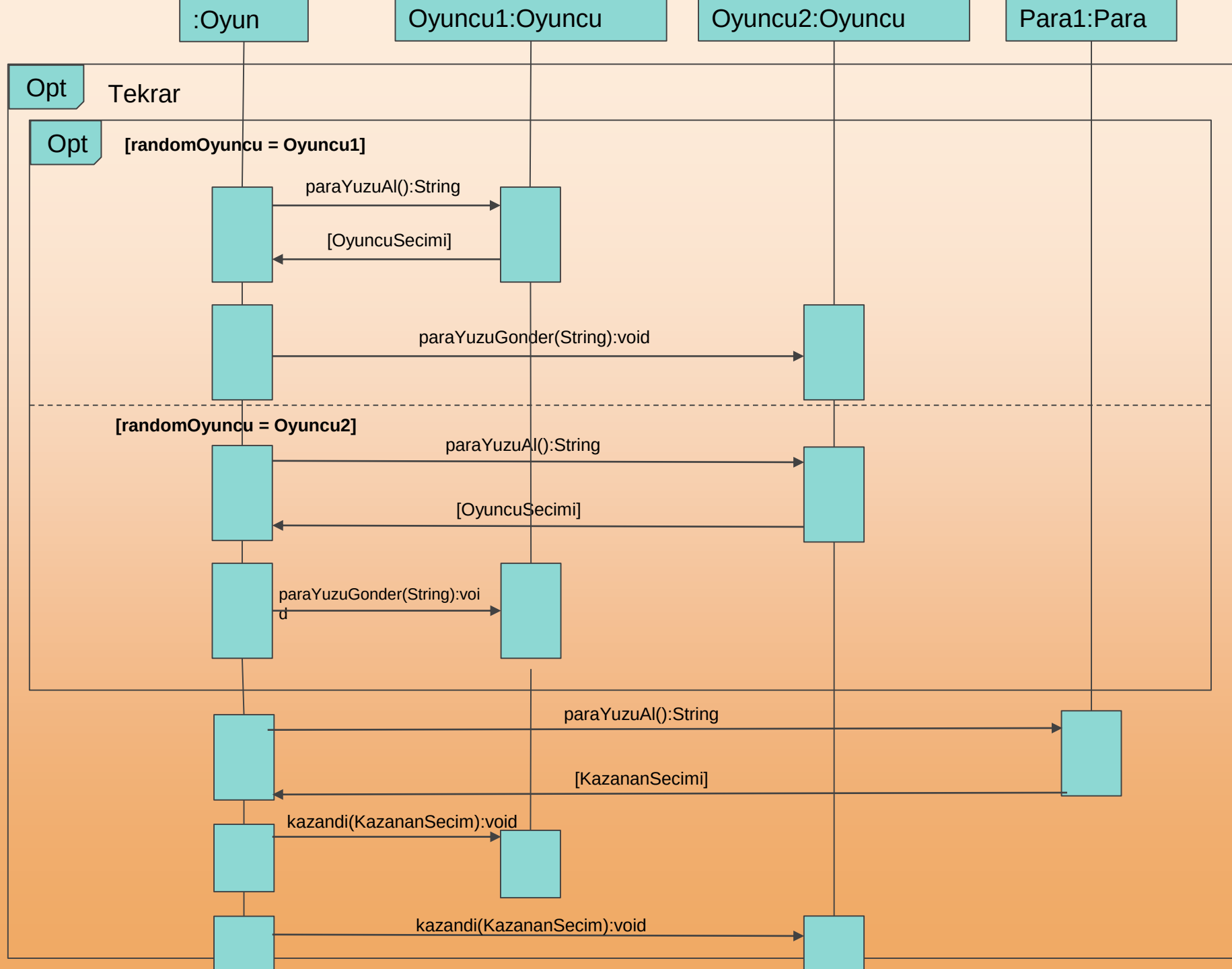
Çalıştırma Adımları

1) **Bir oyuncu tahminini yapmak için oyun tarafından random olarak seçilir**
2) **Seçilen Oyuncu Tahminini yapar**
3) **Diğer oyuncuya oyun kalan tahmini gönderir**
4) **Para atılır ve sonuç oyunculara bildirilir.**
5) **Tekrar oynansın mı diye sorulur.**

UML Nesne Modeli



UML Sıralı Diyagram (Sequence Diagram)



Ve son olarak
Bu kodla ilgili ne düşünüyorsunuz?

```
#include <iostream>

using namespace std;
int temp1;
int temp2;
int temp3;

int firstvalue(){
 temp1 = 0;
 temp2 = 0;
 temp3 = 0;
}

int hesapla(int in1, int in2, int in3){
 int temp1, temp2;
 temp1 = in1*in1 - 4*in2*in3;
 temp2 = (-in1 - sqrt(temp1))/2*in2;
 temp3 = (in1 - sqrt(temp1))/2*in2;
 return 0;
}

int main()
{
   cout << "Hello world!" << hesapla(1, 2, 4); << endl;
   return 0;
}
```

Teşekkürler